1. A

we can get the w^* when its first derivative = 0

$$\Rightarrow \frac{1}{N} \cdot \sum_{n=1}^{N} 2(wx_n - y_n)x_n + \frac{\lambda}{N} 2w = 0$$

$$w\sum_{n=1}^{N} x_n^2 - \sum_{n=1}^{N} x_n y_n + w\lambda = 0$$

$$w\left(\sum_{n=1}^{N} x_n^2 + \lambda\right) = \sum_{n=1}^{N} x_n y_n$$

$$w = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2 + \lambda} = w^* \qquad C = (w^*)^2 = \left(\frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2 + \lambda}\right)^2 \text{ ※}$$

2. B

i. $u = 0$

ii. $\tilde{\phi}(x) = \Gamma^{-1}x$

$$\min_{\tilde{w} \in R^{d+1}} \frac{1}{N}\sum_{n=1}^{N}(\tilde{w}^T\Gamma^{-1}x_n - y_n)^2 + \frac{\lambda}{N}(\tilde{w}^T w) = \min_{w \in R^{d+1}} \frac{1}{N}\sum_{n=1}^{N}(w^T x_n - y_n)^2 + \frac{\lambda}{N}\Omega(w)$$

$$\Rightarrow \tilde{w}^T\Gamma^{-1}x_n = w^T x_n$$

$$\tilde{w}^T\Gamma^{-1} = w^T \Rightarrow \tilde{w}^T = w^T\Gamma^T \Rightarrow \tilde{w} = \Gamma w$$

$$\Rightarrow \Omega(w) = \tilde{w}^T\tilde{w} = w^T\Gamma^T\cdot\Gamma w = w^T\Gamma^2 w$$

3. A

assume all $y_n = 1$

$\Rightarrow err(w, x, y) = \ln(1 + e^{-w^T x})$

$\Rightarrow err_{smooth}(w, x, y) = \frac{\alpha}{2} \ln(1 + e^{w^T x}) + (1 - \frac{\alpha}{2}) \ln(1 + e^{-w^T x})$

solve error smooth

$$\Rightarrow \frac{1}{N} \sum_{n=1}^{N} (1 - \frac{\alpha}{2}) \ln(1 + e^{-w^T x_n}) + \frac{\alpha}{2} \ln(1 + e^{w^T x_n})$$

$$= \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-w^T x_n}) + \frac{\alpha}{2} \ln\left(\frac{1 + e^{w^T x_n}}{1 + e^{-w^T x_n}}\right) = \frac{1}{N} \left[\sum_{n=1}^{N} \ln(1 + e^{-w^T x_n}) + \frac{\alpha}{2} \cdot w^T x_n \right]$$

Solve error

$$\Rightarrow \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-w^T x_n}) + \frac{1}{N} \cdot \frac{\alpha}{1 - \alpha} \cdot \sum_{n=1}^{N} \left[(\frac{1}{2}) \cdot \ln\left(\frac{1 + e^{-w^T x}}{2}\right) + (\frac{1}{2}) \cdot \ln\left(\frac{1 + e^{w^T x}}{2}\right) \right]$$

$$= \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-w^T x_n}) + \frac{1}{N} \cdot \frac{\alpha}{1 - \alpha} \cdot \sum_{n=1}^{N} \left[\ln(1 + e^{-w^T x}) + \frac{1}{2} \cdot (w^T x) \right]$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left[\frac{1}{1 - \alpha} \ln(1 + e^{-w^T x_n}) + \frac{1}{1 - \alpha} \cdot \frac{\alpha}{2} \cdot (w^T x) \right]$$

$$= \frac{1}{N} \cdot \frac{1}{1 - \alpha} \cdot \sum_{n=1}^{N} \left[\ln(1 + e^{-w^T x_n}) + \frac{\alpha}{2} (w^T x) \right]$$

// $y = 1$

// $h(x) = \frac{1}{1 + e^{-w^T x}}$

$y = -1$

$(1 - h(x)) = \frac{1}{1 + e^{w^T x}}$

// if we want to get the optimal w for two function

we will need the w be the answer that makes the first derivative equals 0

// since we need first derivative and equals 0

$\Rightarrow$ the coefficient $\frac{1}{1 - \alpha}$ doesn't matter

$\Rightarrow$ other in the equation are the same

$\Rightarrow \Omega(w, x) = D_{KL}(P_u \| P_h)$ ※

4. D

$$E_{loocv} = \frac{1}{3}\left[\left(\frac{y_1+y_2}{2}-1\right)^2+\left(\frac{y_1+1}{2}-y_2\right)^2+\left(\frac{y_2+1}{2}-y_1\right)^2\right] \le \frac{1}{3}$$

$$\Rightarrow y_1^2+y_2^2-y_1y_2-y_1-y_2+\frac{1}{3} \le 0$$

∵ this is a rotated ellipse let $A=1$ $B=1$ $H=-\frac{1}{2}$ $G=-\frac{1}{2}$ $F=-\frac{1}{2}$ $C=\frac{1}{3}$

$$L=\frac{1}{4}+\frac{1}{4}+\frac{1}{12}-\frac{1}{3}+2\cdot\frac{1}{8}=\frac{1}{2}$$

$$\text{area} = \frac{\pi L}{(AB-H^2)^{\frac{3}{2}}} = \frac{\frac{1}{2}\pi}{(1-\frac{1}{4})^{\frac{3}{2}}} = \frac{4\pi}{3\sqrt{3}}$$

$$\therefore \text{the probability} = \frac{\frac{4\pi}{3\sqrt{3}}}{2\times 2} = \frac{\pi}{3\sqrt{3}}$$

(∵ y is uniformly generated between $[0,2]$)
and the probability depends only on y_1 and y_2

5. D

$$\underset{D_{val}\sim p^k}{Var}[E_{val}(h)] = \underset{D_{val}\sim p^k}{Var}\left[\frac{1}{k}\sum_{n=1}^{k} err(h(x_n),y_n)\right] = \frac{1}{k^2}\underset{D_{val}\sim p^k}{Var}\left[\sum_{n=1}^{k} err(h(x_n),y_n)\right]$$

$$= \frac{1}{k^2}\sum_{n=1}^{k}\underset{D_{val}\sim p^k}{Var}[err(h(x_n),y_n)] = \frac{1}{k^2}\cdot k\cdot \underset{(x,y)\sim p}{Var}[err(h(x),y)]$$

$$= \frac{1}{k}\underset{(x,y)\sim p}{Var}[err(h(x),y)] \quad \#$$

6. D

case 1:
pick out the positive example

$\Rightarrow N-1$ positive ; N negative in training data

$\Rightarrow A_{maj}$ = negative $\Rightarrow$ check on the pick out one is wrong

case 2:
pick out the negative example

$\Rightarrow N-1$ negative ; N positive in training data

$\Rightarrow A_{maj}$ = positive $\Rightarrow$ check on the pick out one is wrong

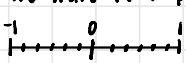
$$E_{loocv} = \frac{1}{2N}\sum e_n = \frac{1}{2N}\underbrace{(N+N)} = 1 \quad \#$$

(∵ N positive; N negative
each of them is wrong when picked out)

7. C

for the decision stump,

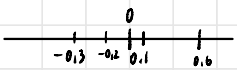
we have N+1 possible hypothesis



∵ decision dump will be two datas' average, the data is based on sign(x)

∴ the only data that may be affected are the two closest to 0 (one for positive and one for negative)

ex.



∃ decision stump result (pick out 0.4) $= \frac{0.6-0.2}{2} = 0.2 > 0.1$

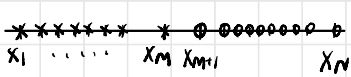
∃ 0.1 will be wrong

the same for the negative side

$\exists E_{loocv} \leq \frac{2}{N}$ ※

(∵ at most the two are all wrongly detected)

8. E



∵ margin is symmetry by the line ∃ margin space = 2 × margin

(O: y=+1 ; X: y=-1)

∴ we want the whole space between two different label data be the margin space

$\exists |x_M - x_{M+1}|$ = margin space = 2 · margin

$\exists$ margin $= \frac{|x_M - x_{M+1}|}{2} = \frac{(x_{M+1} - x_M)}{2}$ ($\because x_{M+1} > x_M$)

9. E

$w = [w_0, w_1]$

$$\begin{cases} w_1 + b \geq 1 & -① \\ -2w_0 - b \geq 1126 & -② \\ -w_1 + b \geq 1 & -③ \\ b \geq 1 & -④ \end{cases}$$

④

$\exists b \geq 1$

① - ③

$2w_1 \geq 0 \quad \exists w_1 \geq 0$

② + ③

$-2w_0 \geq 1127 \quad \exists w_0 \leq -\frac{1127}{2}$

$\exists$ the optimal will be each of its bound

$\exists w = (-\frac{1127}{2}, 0)$; $b = 1$ ※

10. D

we want $E_{in}(\hat{h}) = 0$

that is for every (x_k, y_k) $(1 \le k \le N)$

we need to have $y_k \cdot \text{sign}\left(\sum_{n=1}^{N} y_n \alpha_n K(x_n, x_k) + b\right) > 0$

// $\alpha = 1$ and $b = 0$

$\Rightarrow y_k \cdot \text{sign}\left(\sum_{n=1}^{N} y_n \cdot e^{-r\|x_n - x_k\|^2}\right) > 0$

since we want to find a lower bound of r

it means that we want to have as many $e^{-r|x-x_k|^2}$ as possible

↳ which says that we have to make all the data with same label

$\Rightarrow$

if $y_k = +1$, other $y_n = -1$: $\sum_{n=1}^{N} y_n e^{-r|x-x_k|^2} > 0$ | if $y_k = -1$, other $y_n = +1$: $\sum_{n=1}^{N} y_n \cdot e^{-r|x-x_k|^2} < 0$

$\Rightarrow$ we need to solve $1 - (N-1) e^{-r|x-x_k|^2} > 0$

$1 - (N-1) e^{-r|x-x_k|^2} \ge 1 - (N-1) e^{-r\xi^2} > 0$

$(N-1) e^{-r\xi^2} < 1 \Rightarrow e^{-r\xi^2} < \frac{1}{N-1} \Rightarrow -r\xi^2 < \ln\left(\frac{1}{N-1}\right)$

$\Rightarrow r\xi^2 > \ln(N-1) \Rightarrow r > \frac{\ln(N-1)}{\xi^2}$ //

17.

D

$$d = \sqrt{\|\phi(x) - \phi(x')\|^2}$$

$$= \sqrt{\phi(x)\phi(x) - 2\phi(x)\phi(x') + \phi(x')\phi(x')}$$

$$= \sqrt{K(x,x) - 2K(x,x') + K(x',x')}$$

$$= \sqrt{e^0 - 2e^{-r|x-x'|^2} + e^0} = \sqrt{2 - 2e^{-r|x-x'|^2}} \leq \sqrt{2} = 1.4$$

$\exists$ tightest upperbound $= \sqrt{2} < 1.5$ //

HTML HW4 Coding part

資工二B10705005 陳思如

import libraries:

```
import numpy as np
from numpy import loadtxt
import random
from random import randint
import statistics
from statistics import mode
from statistics import mean
import math
from liblinear.liblinearutil import*
import itertools
from itertools import combinations
from itertools import combinations_with_replacement
```

function definition:

```
#function for transformation of x
def fourth_order(x):
    n = len(x)
    num = list(itertools.chain(range(0, n)))

    res = list(combinations_with_replacement(num, 2))
    for i in res:
      r = 1
      for j in i:
        r *= x[j]
      x = np.append(x, r, axis = None)

    res = list(combinations_with_replacement(num, 3))
    for i in res:
      r = 1
      for j in i:
        r *= x[j]
      x = np.append(x, r, axis = None)

    res = list(combinations_with_replacement(num, 4))
    for i in res:
      r = 1
      for j in i:
        r *= x[j]
      x = np.append(x, r, axis = None)

    x = np.insert(x, 0, 1, axis = None)
    return x

#function for calculate 0/1 error
def calE(w, x, ans):
    Eout = float(0)
    for i in range(len(x)):
```

```
        sum = 0
        for j in range(len(x[i])):
            sum += w[j]*x[i][j]
        if(sum*ans[i] <= 0):
            Eout += 1
    Eout /= float(len(x))
    return Eout
```

prepare data:

```
#data prepare
file = open('train.txt', 'r')
data = []
for line in file:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data.append(numbers)

x = np.array(data)
y = x[:, -1]
x = x[:, :-1]
fourth = fourth_order(x[0])
xtrain = np.empty((0, len(fourth)))
for i in x:
    xtrain = np.vstack([xtrain, fourth_order(i)])


file2 = open('test.txt', 'r')
data2 = []
for line in file2:
    number_strings = line.split()
    numbers = [float(n) for n in number_strings]
    data2.append(numbers)

x_test = np.array(data2)
y_test = x_test[:, -1]
x_test = x_test[:, :-1]
fourth = fourth_order(x_test[0])
xtest = np.empty((0, len(fourth)))
for i in x_test:
    xtest = np.vstack([xtest, fourth_order(i)])

lamb = np.array([-6, -3, 0, 3, 6])
min_Eout = 2
min_Ein = 2
bestout = 0
bestin = 0
```

Q12 D

Q13 C

```
#Q12 Q13
min_Eout = 2
min_Ein = 2
for la in lamb:
    print("lambda = ", la, "::")
    ld = math.pow(10, la)
    c = 1/(2*ld)
    w = train(problem(y, xtrain), parameter('-q -s 0 -c ' + str(c) + ' -e
0.000001')).get_decfun()[0]
    eout = calE(w, xtest, y_test)
    print("eout = ", eout)
    if(eout <= min_Eout):
        min_Eout = eout
        bestout = la

    ein = calE(w, xtrain, y)
    print("ein = ", ein)
    if(ein <= min_Ein):
      min_Ein = ein
      bestin = la

print("min Eout with lambda = ",bestout)
print("min Ein with lambda = ",bestin)
```

Q14 D

Q15 C

Q16 B

```
#Q14 Q15 Q16
valla = []
Eout = []
Eout_ = []
for i in range(256):
  min_Eval = 2
  bestw = []
  bestla = 0
  randomlist = random.sample(range(0, len(xtrain)), 120)
  x_tr = np.empty((0, len(xtrain[0])))
  y_tr = np.empty((0, 1))
  x_val = np.empty((0, len(xtrain[0])))
  y_val = np.empty((0, 1))
  for i in range(0, len(x)):
    if i in randomlist:
      x_tr = np.vstack([x_tr, xtrain[i]])
      y_tr = np.vstack([y_tr, y[i]])
    else:
      x_val = np.vstack([x_val, xtrain[i]])
      y_val = np.vstack([y_val, y[i]])
  y_tr = y_tr.reshape(len(y_tr),)
  y_val = y_val.reshape(len(y_val),)
  for la in lamb:
    ld = math.pow(10, la)
    c = 1/(2*ld)
    w = train(problem(y_tr, x_tr), parameter('-q -s 0 -c ' + str(c) + ' -e
0.000001')).get_decfun()[0]
    eval = calE(w, x_val, y_val)

    if(eval <= min_Eval):
        min_Eval = eval
        bestla = la
        bestw = w

  valla.append(bestla)
  Eout.append(calE(bestw, xtest, y_test))

  ld = math.pow(10, bestla)
  c = 1/(2*ld)
  w_ = train(problem(y, xtrain), parameter('-q -s 0 -c ' + str(c) + ' -e
0.000001')).get_decfun()[0]
  Eout_.append(calE(w_, xtest, y_test))

print(mode(valla))
print(mean(Eout))
print(mean(Eout_))
```

Q17 A

```
#Q17
Ecv = []
for i in range(256):
  min_Ecv = 2
  bestw = []
  bestla = 0
  randomlist = list(range(0, len(x)))
  random.shuffle(randomlist)

  for la in lamb:
    #print("lambda = ", la, "::")
    ld = math.pow(10, la)
    c = 1/(2*ld)
    ecv = float(0)
    for k in range(5):
      x_tr = np.empty((0, len(xtrain[0])))
      y_tr = np.empty((0, 1))
      x_val = np.empty((0, len(xtrain[0])))
      y_val = np.empty((0, 1))
      for j in range(0, len(x)):
        if j < 40*(k+1) and j >= (40)*k:
          x_val = np.vstack([x_val, xtrain[j]])
          y_val = np.vstack([y_val, y[j]])
        else:
          x_tr = np.vstack([x_tr, xtrain[j]])
          y_tr = np.vstack([y_tr, y[j]])
      y_tr = y_tr.reshape(len(y_tr),)
      y_val = y_val.reshape(len(y_val),)

      w = train(problem(y_tr, x_tr), parameter('-q -s 0 -c ' + str(c) + ' -e
0.000001')).get_decfun()[0]
      ecv += float(calE(w, x_val, y_val))
    ecv /= float(len(x)/40)

    if(ecv <= min_Ecv):
        min_Ecv = ecv
        bestla = la
        bestw = w

  Ecv.append(min_Ecv)

print(mean(Ecv))
```

Q18 C

Q19 E

```
#Q18 Q19
bestw_l1 = []
min_Eout = 2
for la in lamb:
    print("lambda = ", la, "::")
    ld = math.pow(10, la)
    c = 1/(ld)
    w = train(problem(y, xtrain), parameter('-q -s 6 -c ' + str(c) + ' -e
0.000001')).get_decfun()[0]
    eout = calE(w, xtest, y_test)
    print("eout = ", eout)
    if(eout <= min_Eout):
        min_Eout = eout
        bestout = la
        bestw_l1 = w

print("min Eout with lambda = ",bestout)
cnt = 0
for i in bestw_l1:
  if abs(i) <= math.pow(10, -6):
    cnt += 1
print(cnt)
```

Q20 A

```
#Q20
bestw_l2 = []
min_Eout = 2
for la in lamb:
    print("lambda = ", la, "::")
    ld = math.pow(10, la)
    c = 1/(2*ld)
    w = train(problem(y, xtrain), parameter('-q -s 0 -c ' + str(c) + ' -e
0.000001')).get_decfun()[0]
    eout = calE(w, xtest, y_test)
    print("eout = ", eout)
    if(eout <= min_Eout):
      min_Eout = eout
      bestout = la
      bestw_l2 = w

print("min Eout with lambda = ",bestout)
cnt = 0
for i in bestw_l2:
  if abs(i) <= math.pow(10, -6):
    cnt += 1
print(cnt)
```