Fin :

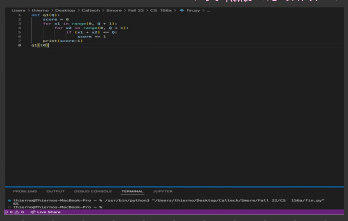
(1)[e]

We know that Z is a linear combination of $x_1, x_2 \in X$. for example as in lec 15, slide 4 $x=(x_1, x_2)$ then $\Phi(x) = z = (x_0, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3) = (1, x_2, x_1, x_1x_2, x_1^2, x_2^2)$

Now repeat the above power for $Q=10$ on the $(x_1, x_2) \in X$.

$z = (x_0^{10}, x_1x_2^9, x_1^2x_2^8, x_1^3x_2^7, x_1^4x_2^6, \ldots, x_1^{10}x_2^0)$

resulting in a 65 dimension on Z space for all linear combinations of $x_1^i x_2^j$ as I programmed from because it was too much to count I subtracted 1 from $Score$ bcz we are excluding $x_1^0 x_2^0$. Hence the correct answer is [e].



(2) [d]

[a] This is false because in this case we will have the same hypothesis for any dataset D, thus the average hypothesis will just be our single hypothesis in $\mathcal{H}$ thus $\bar{g} \in \mathcal{H}$.

[b] This is false because in this case any chosen hypothesis g^D for some dataset D will be a constant $C \in \mathbb{R}$, thus our average hypothesis, $\bar{g}$, would be the average of these real valued numbers hence which will be inherently a subset of real valued numbers hence $\bar{g} \in \mathcal{H}$ as $\mathcal{H}$ is a set of real numbers.

[c] This is false because in this case any chosen hypothesis g^D for any given dataset will be a linear combination of $mx+b$ where m & b are real-valued coefficients hence the average hypothesis would be a linear model in the form $\bar{m}x+\bar{b}$ where $\bar{m}$ & $\bar{b}$ are real valued constants hence contained in our $\mathcal{H}$ of linear models hence $\bar{g} \in \mathcal{H}$.

[d] This is true because in this case our hypothesis set relies on the Sigmoid function mapping from 0 to 1 or tangent mapping from -1 to 1 as seen in lecture 9. However, the average hypothesis $\bar{g}$ of D might not necessarily be a Sigmoid or tanh as these functions are not linear hence $\bar{g} \notin \mathcal{H}$.

[e] false bcz d is correct.

(3) [d]

[a] True because overfitting occurs when a hypothesis is picked with a lower E_{in} and higher E_{out} as seen in lecture 11 hence we need atleast two hypotheses with varying E_{in} which leads to overfitting.

[b] True bcz overfitting is a result of our choosing hypotheses with varying E_{out} as shown in lecture 11 bcz we are choosing an incorrect hypothesis with a higher E_{out} due to a low E_{in} (also explained in lec 11 slide 12)

[c] True again by lecture 11, we know we need two hypotheses with different values for $(E_{out} - E_{in})$ because then we'd have overfitting bcz to one of them being an inferior hypothesis as a result of a low E_{in} & high E_{out}.

[d] False because consider the below examples.

Let $\mathcal{H}_1$ & $\mathcal{H}_2$ be our hypothesis set & K_1, K_2 be our second with the below characteristics:

set 1: $E_{in}(h_1) = 0.1$ & $E_{out}(h_1) = 0.5 \therefore E_{out} - E_{in} = 0.4$, $\min E_{in}(\mathcal{H}_2) = 0.8$ & $E_{out}(h_2) = 0.9 \therefore E_{out} - E_{in} = 0.1$

Here we do not have overfitting due to high E_{in} in $\mathcal{H}_2$.

set 2: $E_{in}(K_1) = 0.1$ & $E_{out}(K_1) = 0.5 \therefore E_{out} - E_{in} = 0.4$, $\min E_{in}(K_2) = 0.1$ & $E_{out}(K_2) = 0.2 \therefore E_{out} - E_{in} = 0.1$

We have overfitting due to low E_{in} in $\mathcal{H}_2$.

Choice [d] also contradicts how we determine our overfitting, $E_{out}(g_1) - E_{out}(g_n)$ where g_1 & g_n are our two hypotheses, as seen in lecture 11 slide 12 hence making [d] false hence the correct answer

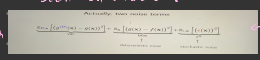
[e] True because we need atleast two hypotheses to determine overfitting as discussed in lecture 11.

(4) [d]

[a] False because the two can occur together as seen in lecture 11 slide 20.

[b] False by the definition of deterministic noise from lecture 11 stating that it depends on our hypothesis set as it is our bias as seen on slide 20.

[c] False by the definition of deterministic noise from lecture 11 stating that it depends on our target function as it is our bias as seen on slide 20.



[d] True as seen by the above lecture equation but also by the definition of stochastic noise from lecture 11 as it's the presence of random error in our data set.

[e] False because stochastic noise depends on our dataset which is determined by our target function as seen in the above equation.

(5) [a]

Looking at lecture 12, slide 7 we see that we derived W_{lin} by minimizing $E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}(w^T z_n - y_n)^2$ which resulted in $w_{lin} = (Z^T Z)^{-1} Z^T y$.

In the above question W_{lin} is subject to the constraint $W_{lin}^T \Gamma^T \Gamma W_{lin} \leq C$. looking at the problem statement we see that it defines W_{reg} the same way we derived W_{lin} from lecture thus substituting W_{lin} with W_{reg} we get that W_{reg} also satisfies the constraint $W_{lin}^T \Gamma^T \Gamma W_{lin} \leq C$ hence leading to the conclusion that $W_{reg} = W_{lin}$ hence the correct answer is [a] as both satisfy the constraint.

(6) [b]

[a] False as we can't write soft-order constraints as hard order-constraints due to the latter having no upper bound.

[b] Correct because looking at lecture 12, slide 8 we know that our soft-order constraint is found by minimizing:

$\frac{1}{N}(Zw-y)^T(Zw-y)$ subject to $w^T w \leq C$ with $w_{reg} = (Z^T Z + \lambda I)^{-1} Z^T y$ as a solution with λ being our regularization factor.

Now we know this could be translated into Augment error as done in lecture 12, slide 9 as our soft-order constraint was translated into $\min E_{in}(w) + \frac{\lambda}{N} w^T w$ which is our augmented error.

Hence as our soft-order constraint minimizes Augmented thus it can be translated into augmented error.

[c] False because the value of the VC dimension does not determine or soft-order constraint as it is found by minimizing Augmented as seen in lecture 12 slides 9-12.

[d] False because it increases E_{in} while decreasing E_{out}.

[e] False bcz b is correct.

(7) [d]

```python
train = numpy.loadtxt('./features.train')
test = numpy.loadtxt('./features.test')
X_train = []
X_test = test[:, 0]
for d1 in train:
    X_train.append([1, d1[0], d1[1]])
digits_train = train[:, 0]
digits_test = test[:, 0]

def make_binary(digit_class, digits):
    scores = []
    for score in digits:
        if (score != digit_class):
            scores.append(-1)
        else:
            scores.append(1)
    return numpy.array(scores)

def get_weight_reg(k, data, binary_train):
    Z = numpy.array(data)
    dagger = (numpy.dot(Z.T, Z) + numpy.dot(numpy.identity(3), k))
    w =  numpy.dot(numpy.dot(numpy.linalg.inv(dagger), Z.T), binary_train)
    return w

def Ein_lin(d_classifier, k):
    summ_err = 0
    binary_train = make_binary(d_classifier, digits_train)
    Z = X_train
    w = get_weight_reg(k, Z, binary_train)
    for d,s in zip(Z, binary_train):
        if (s != numpy.sign(numpy.dot(w, d))):
            summ_err += 1
    return summ_err/len(Z)

def q7():
    print("5 versus all our Ein is: ", Ein_lin(5, 1))
    print("6 versus all our Ein is: ", Ein_lin(6, 1))
    print("7 versus all our Ein is: ", Ein_lin(7, 1))
    print("8 versus all our Ein is: ", Ein_lin(8, 1))
    print("9 versus all our Ein is: ", Ein_lin(9, 1))

q7()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/fin.py"
5 versus all our Ein is:  0.07625840076807022
6 versus all our Ein is:  0.09107118365107666
7 versus all our Ein is:  0.08846523110684405
8 versus all our Ein is:  0.07433822552016199
9 versus all our Ein is:  0.08832807570977919
thierno@Thiernos-MacBook-Pro CS  156a %
```

Hence according to the above code the correct answer is 8 v all hence choice [d]

(8) [b]

```python
digits_train = train[:, 0]
X_train = train[:, 1:]
digits_test = test[:, 0]
X_test = test[:, 1:]

def make_binary(digit_class, digits):...
def transform(x1, x2):
    return [1, x1, x2, x1 * x2, x1 ** 2, x2 ** 2]

def get_weight_reg(k, data, binary_train, Transform):
    i = 2 if (Transform == False) else 6
    Z = numpy.array(data)
    dagger = (numpy.dot(Z.T, Z) + numpy.dot(numpy.identity(i), k))
    w =  numpy.dot(numpy.dot(numpy.linalg.inv(dagger), Z.T), binary_train)
    return w

def Ein_lin(d_classifier, k, Transform):...

def Eout_lin(d_classifier, k, Transform):
    binary_test = make_binary(d_classifier, digits_test)
    binary_train = make_binary(d_classifier, digits_train)
    summ_err= 0
    if (Transform == False):
        Z_Test = X_test
        Z_Train = X_train
    else:
        Z_Train = []
        Z_Test = []
        for point1 in X_train:
            Z_Train.append(transform(point1[0], point1[1]))
        for point1 in X_test:
            Z_Test.append(transform(point1[0], point1[1]))
    w = get_weight_reg(k, Z_Train, binary_train, Transform)
    for d, s in zip(Z_Test, binary_test):
        if (s != numpy.sign(numpy.dot(w, d))):
            summ_err += 1
    return summ_err / len(Z_Test)

def q7():...
def q8():
    print("0 versus all our Eout is: ", Eout_lin(0, 1, True))
    print("1 versus all our Eout is: ", Eout_lin(1, 1, True))
    print("2 versus all our Eout is: ", Eout_lin(2, 1, True))
    print("3 versus all our Eout is: ", Eout_lin(3, 1, True))
    print("4 versus all our Eout is: ", Eout_lin(4, 1, True))
q8()
```

```
PROBLEMS  11  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/fin.py"
0 versus all our Eout is:  0.10662680617837568
1 versus all our Eout is:  0.021923268560003986
2 versus all our Eout is:  0.09865478052017937
3 versus all our Eout is:  0.08271051320378675
4 versus all our Eout is:  0.09965122072745392
thierno@Thiernos-MacBook-Pro CS  156a %
```

Hence according to the above code the correct answer is 1 v all hence [b]

All helper functions used are the same throughout all code hence see original definition

(9) [e]

fin.py > q9

```python
def make_binary(digit_class, digits):
    scores = []
    for score in digits:
        if (score != digit_class):
            scores.append(-1)
        else:
            scores.append(1)
    return numpy.array(scores)
def transform(x1, x2):
    return [1, x1, x2, x1 * x2, x1 ** 2, x2 ** 2]

def get_weight_reg(k, data, binary_train, Transform):
    i = 2 if (Transform == False) else 6
    Z = numpy.array(data)
    dagger = (numpy.dot(Z.T, Z) + numpy.dot(numpy.identity(i), k))
    w =  numpy.dot(numpy.dot(numpy.linalg.inv(dagger), Z.T), binary_train)
    return w

def Ein_lin(d_classifier, k, Transform):
    summ_err = 0
    binary_train = make_binary(d_classifier, digits_train)
    if (Transform == False):
        Z = X_train
    else:
        Z = []
        for point1 in X_train:
            Z.append(transform(point1[0], point1[1]))
    w = get_weight_reg(k, Z, binary_train, Transform)
    for d,s in zip(Z, binary_train):
        if (s != numpy.sign(numpy.dot(w, d))):
            summ_err += 1
    return summ_err/len(Z)

def Eout_lin(d_classifier, k, Transform):
    binary_test = make_binary(d_classifier, digits_test)
    binary_train = make_binary(d_classifier, digits_train)
    summ_err= 0
    if (Transform == False):
        Z_Test = X_test
        Z_Train = X_train
    else:
        Z_Train = []
        Z_Test = []
        for point1 in X_train:
            Z_Train.append(transform(point1[0], point1[1]))
        for point1 in X_test:
            Z_Test.append(transform(point1[0], point1[1]))
    w = get_weight_reg(k, Z_Train, binary_train, Transform)
    for d, s in zip(Z_Test, binary_test):
        if (s != numpy.sign(numpy.dot(w, d))):
            summ_err += 1
    return summ_err / len(Z_Test)
```

All Helper functions Used are the Same throughtout all code hence See original defininton

```python
> def q8():...
def q9():
    print('Num:     (Ein, Eout) With Transformation:                          (Ein, Eout) Without Transformation:      ')
    print('0  :     (',Ein_lin(0, 1, True),' , ', Eout_lin(0, 1, True),')  |     (',Ein_lin(0, 1, False),' , ', Eout_lin(0, 1, False))
    print('1  :     (',Ein_lin(1, 1, True),' , ', Eout_lin(1, 1, True),') |     (',Ein_lin(1, 1, False),'  , ',Eout_lin(1, 1, False))
    print('2  :     (',Ein_lin(2, 1, True),' , ', Eout_lin(2, 1, True),')  |     (',Ein_lin(2, 1, False),' , ', Eout_lin(2, 1, False))
    print('3  :     (',Ein_lin(3, 1, True),' , ', Eout_lin(3, 1, True),')  |     (',Ein_lin(3, 1, False),' , ', Eout_lin(3, 1, False))
    print('4  :     (',Ein_lin(4, 1, True),' , ', Eout_lin(4, 1, True),')  |     (',Ein_lin(4, 1, False),' , ', Eout_lin(4, 1, False))
    print('5  :     (',Ein_lin(5, 1, True),' , ', Eout_lin(5, 1, True),')  |     (',Ein_lin(5, 1, False),' , ', Eout_lin(5, 1, False))
    print('6  :     (',Ein_lin(6, 1, True),' , ', Eout_lin(6, 1, True),')  |     (',Ein_lin(6, 1, False),' , ', Eout_lin(6, 1, False))
    print('7  :     (',Ein_lin(7, 1, True),' , ', Eout_lin(7, 1, True),')  |     (',Ein_lin(7, 1, False),' , ', Eout_lin(7, 1, False))
    print('8  :     (',Ein_lin(8, 1, True),' , ', Eout_lin(8, 1, True),')  |     (',Ein_lin(8, 1, False),' , ', Eout_lin(8, 1, False))
    print('9  :     (',Ein_lin(9, 1, True),' , ', Eout_lin(9, 1, True),')  |     (',Ein_lin(9, 1, False),' , ', Eout_lin(9, 1, False))
    print(abs(Eout_lin(5, 1, True) - Eout_lin(5, 1, False))/ Eout_lin(5, 1, True) * 100)
q9()
> def q10():...
```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/fin.py"
Num:    (Ein, Eout) With Transformation:                     (Ein, Eout) Without Transformation:
0  :    ( 0.10231792621039638  ,  0.10662680617837568 )  |    ( 0.22946097928953504  ,  0.22770303936223218
1  :    ( 0.012343985735838706  ,  0.02192326856003986 ) |    ( 0.137704018653134    ,  0.1310413552566019
2  :    ( 0.10026059525442327  ,  0.09865470852017937 )  |    ( 0.10026059525442327  ,  0.09865470852017937
3  :    ( 0.09024825126868742  ,  0.08271051320378675 )  |    ( 0.09024825126868742  ,  0.08271051320378675
4  :    ( 0.08942531888629818  ,  0.09965122072745392 )  |    ( 0.08942531888629818  ,  0.09965122072745392
5  :    ( 0.07625840076807022  ,  0.07922272047832586 )  |    ( 0.07625840076807022  ,  0.07972097658196313
6  :    ( 0.09107118365107666  ,  0.08470353761833582 )  |    ( 0.09107118365107666  ,  0.08470353761833582
7  :    ( 0.08846523110684405  ,  0.07324364723467862 )  |    ( 0.08846523110684405  ,  0.07324364723467862
8  :    ( 0.07433822520916199  ,  0.08271051320378675 )  |    ( 0.07433822520916199  ,  0.08271051320378675
9  :    ( 0.08832807570977919  ,  0.08819133034379671 )  |    ( 0.08832807570977919  ,  0.08819133034379671
0.6289308176100608
thierno@Thiernos-MacBook-Pro CS  156a %
```

[a]&[b] are false bcz of 2vall

[c]&[d] are false bcz of 5vall

0.63% improvement of out-of sample performace for 5 V all which is less than 5% b/z of transform.

hence according to the above code [e] is correct

(10) [a]

```python
import numpy
import random
from sklearn import svm
train = numpy.loadtxt('./features.train')
test = numpy.loadtxt('./features.test')

def one_v_one(data, num1, num2):
    new_data = []
    for thing in data:
        if thing[0] != num1 and thing[0] != num2:
            continue
        else:
            new_data.append(thing)
    return numpy.array(new_data)
otrain = one_v_one(train, 1, 5)
otest = one_v_one(test, 1, 5)
digits_train = otrain[:, 0]
X_train = otrain[:, 1:]
digits_test = otest[:, 0]
X_test = otest[:, 1:]

```

```python
def make_binary(digit_class, digits):
    scores = []
    for score in digits:
        if (score != digit_class):
            scores.append(-1)
        else:
            scores.append(1)
    return numpy.array(scores)
def transform(x1, x2):
    return [1, x1, x2, x1 * x2, x1 ** 2, x2 ** 2]

def get_weight_reg(k, data, binary_train, Transform):
    i = 2 if (Transform == False) else 6
    Z = numpy.array(data)
    dagger = (numpy.dot(Z.T, Z) + numpy.dot(numpy.identity(i), k))
    w =  numpy.dot(numpy.dot(numpy.linalg.inv(dagger), Z.T), binary_train)
    return w

def Ein_lin(d_classifier, k, Transform):
    summ_err = 0
    binary_train = make_binary(d_classifier, digits_train)
    if (Transform == False):
        Z = X_train
    else:
        Z = []
        for point1 in X_train:
            Z.append(transform(point1[0], point1[1]))
    w = get_weight_reg(k, Z, binary_train, Transform)
    for d,s in zip(Z, binary_train):
        if (s != numpy.sign(numpy.dot(w, d))):
            summ_err += 1
    return summ_err/len(Z)

def Eout_lin(d_classifier, k, Transform):
    binary_test = make_binary(d_classifier, digits_test)
    binary_train = make_binary(d_classifier, digits_train)
    summ_err= 0
    if (Transform == False):
        Z_Test = X_test
        Z_Train = X_train
    else:
        Z_Train = []
        Z_Test = []
        for point1 in X_train:
            Z_Train.append(transform(point1[0], point1[1]))
        for point1 in X_test:
            Z_Test.append(transform(point1[0], point1[1]))
    w = get_weight_reg(k, Z_Train, binary_train, Transform)
    for d, s in zip(Z_Test, binary_test):
        if (s != numpy.sign(numpy.dot(w, d))):
            summ_err += 1
    return summ_err / len(Z_Test)
```

```python
> def q7():…
> def q8():…
> def q9():…
def q10():
    print('lamda = 0.01 (Ein, Eout): ', Ein_lin(1, 0.01, True),',', Eout_lin(1, 0.01, True))
    print('lamda = 1 (Ein, Eout): ', Ein_lin(1, 1, True),',', Eout_lin(1, 1, True))

q10()
```

```
PROBLEMS 11  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

thierno@Thiernos-MacBook-Pro CS  156a % /usr/bin/python3 "/Users/thierno/Desktop/Caltech/Smore/Fall 22/CS  156a/fin.py"
lamda = 0.01 (Ein, Eout):  0.004484304932735426 , 0.02830188679245283
lamda = 1 (Ein, Eout):  0.005124919923126201 , 0.025943396226415096
thierno@Thiernos-MacBook-Pro CS  156a %
```

Hence according to the above code, the correct answer is [a]. as it is the only choice that matches our output

All helper functions used are the same throughout all code hence see original definition