T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4088

Introduction to Machine Learning

Homework #3

150116034 – Enes Garip

Part 1

Q4)

$$\frac{\partial E}{\partial v} = \frac{\partial((v.e^{v}-2ve^{-v})^2)}{\partial v} = 2.(v.e^{v}-2ve^{-v}).(v.e^{v}-2v.e^{-v})'$$

$$= 2.(v.e^{-v}-2ve^{-v}).(e^{v}-(-1).2v.e^{-v})$$

$$= 2.(e^{v}+2.v.e^{-v}).(v.e^{-v}-2ve^{-v})$$

The answer is E.

```python
# HW-5 Question 4, 5 ,6 ,7 Implementation
def hw5q4567():
    # Error calculation
    def E(u, v):
        return (u * math.e ** v - 2 * v * math.e ** (-u)) ** 2

    # Gradient u calculation
    def gradient_u(u, v):
        return 2 * (u * math.e ** v - 2 * v * math.e ** (-u)) * (math.e ** v + 2 * v * math.e ** (-u))

    # Gradient v calculation
    def gradient_v(u, v):
        return 2 * (u * math.e ** v - 2 * v * math.e ** (-u)) * (u * math.e ** v - 2 * math.e ** (-u))

    # Gradient descent calculation
    def gradient_descent(u, v, learning_ratio):
        i = 0
        while True:
            i += 1

            du = gradient_u(u, v)
            dv = gradient_v(u, v)

            u = u - learning_ratio * du
            v = v - learning_ratio * dv

            if E(u, v) < 10 ** (-14):
                break

        return i, u, v

    # Coordinate descent calculation
    def coordinate_descent(u, v, learning_ratio):
        for _ in range(15):
            du = gradient_u(u, v)
            u = u - learning_ratio * du

            dv = gradient_v(u, v)
            v = v - learning_ratio * dv

        return E(u, v)

    # u and v values and learning rate
    u, v = (1, 1)
    learning_rate = 0.1

    iteration, new_u, new_v = gradient_descent(u, v, learning_rate)
    print(f'Iteration = {iteration}')
    print(f'New u = {new_u}\tNew v = {new_v}')

    error = coordinate_descent(u, v, learning_rate)
    print(f'Error after 15 iterations = {error}')
```

In this part, there are functions to calculate the derivations and errors. In gradient descent, it exits if error is below 10^{-14}.

```
Iteration = 10
New u = 0.04473629039778207 New v = 0.023958714099141746
Error after 15 iterations = 0.13981379199615324
Q2) Ein = 0.02857142857142858 Eout = 0.08399999999999996
Q3) Ein = 0.02857142857142858 Eout = 0.07999999999999996
Q4) Ein = 0.37142857142857144 Eout = 0.43600000000000005
Q5)
    k = -2 Ein = 0.02857142857142858 Eout = 0.08399999999999996
    k = -1 Ein = 0.02857142857142858 Eout = 0.05600000000000005
    k = 0 Ein = 0.0 Eout = 0.09199999999999997
    k = 1 Ein = 0.05714285714285716 Eout = 0.124
    k = 2 Ein = 0.19999999999999996 Eout = 0.22799999999999998

Process finished with exit code 0
```

- In question five, the program iterates 10 times. As a result, the answer is D) 10.
- In question six, new u and new v values are 0.04473 and 0.02395 respectively. Therefore, the answer is E) (0.045, 0.024)
- In question seven, coordinate descent function is similar of gradient descent. The result of that function is 0.1398. Therefore, the answer is A) 10^{-1}.

Part 2

-

Part 3

```python
def hw6q23456():
    train_data = np.loadtxt('in.dta')
    test_data = np.loadtxt('out.dta')

    x_train, y_train = train_data[:, :2], train_data[:, 2]
    x_test, y_test = test_data[:, :2], test_data[:, 2]

    n_train = len(x_train)
    x_train = np.array([np.ones(n_train),
                        x_train[:, 0],
                        x_train[:, 1],
                        x_train[:, 0] ** 2,
                        x_train[:, 1] ** 2,
                        x_train[:, 0] * x_train[:, 1],
                        np.absolute(x_train[:, 0] - x_train[:, 1]),
                        np.absolute(x_train[:, 0] + x_train[:, 1])
                        ]).T
    n_test = len(x_test)
    x_test = np.array([np.ones(n_test),
                       x_test[:, 0],
                       x_test[:, 1],
                       x_test[:, 0] ** 2,
                       x_test[:, 1] ** 2,
                       x_test[:, 0] * x_test[:, 1],
                       np.absolute(x_test[:, 0] - x_test[:, 1]),
                       np.absolute(x_test[:, 0] + x_test[:, 1])
                       ]).T

    q2e_in, q2e_out = f_without_regularization(x_train, y_train, x_test, y_test)
    print(f'Q2) Ein = {q2e_in} Eout = {q2e_out}')

    q3e_in, q3e_out = f_with_regularization(-3, x_train, y_train, x_test, y_test)
    print(f'Q3) Ein = {q3e_in} Eout = {q3e_out}')

    q4e_in, q4e_out = f_with_regularization(3, x_train, y_train, x_test, y_test)
    print(f'Q4) Ein = {q4e_in} Eout = {q4e_out}')
    print("Q5) ")
    for i in range(-2, 3):
        q5e_in, q5e_out = f_with_regularization(i, x_train, y_train, x_test, y_test)
        print(f'\tk = {i} Ein = {q5e_in} Eout = {q5e_out}')
```

There are two files named as in.dta and out.dta. After reading these files, I create datasets for test and training. After that, I create formula functions.

```python
def f_without_regularization(train_data_x, train_data_y, test_data_x, test_data_y):
    formula = np.dot(np.linalg.inv(np.dot(train_data_x.T, train_data_x)), train_data_x.T)
    w = np.dot(formula, train_data_y)
    input_data = np.sign(np.dot(train_data_x, w))
    e_in = 1 - np.mean(input_data == train_data_y)
    output_data = np.sign(np.dot(test_data_x, w))
    e_out = 1 - np.mean(output_data == test_data_y)
    return e_in, e_out


def f_with_regularization(x, train_data_x, train_data_y, test_data_x, test_data_y):
    formula = np.dot(np.linalg.inv(np.dot(train_data_x.T, train_data_x) + 10 ** x * np.identity(8)), train_data_x.T)
    w = np.dot(formula, train_data_y)
    input_data = np.sign(np.dot(train_data_x, w))
    e_in = 1 - np.mean(input_data == train_data_y)
    output_data = np.sign(np.dot(test_data_x, w))
    e_out = 1 - np.mean(output_data == test_data_y)
    return e_in, e_out
```

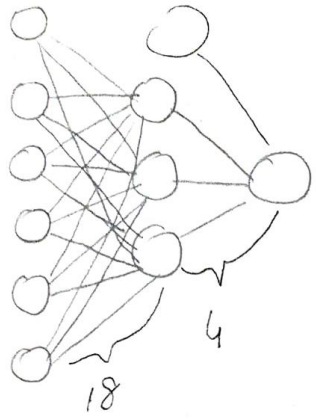
```
Iteration = 10
New u = 0.04473629039778207 New v = 0.023958714099141746
Error after 15 iterations = 0.13981379199615324
Q2) Ein = 0.02857142857142858 Eout = 0.08399999999999996
Q3) Ein = 0.02857142857142858 Eout = 0.07999999999999996
Q4) Ein = 0.37142857142857144 Eout = 0.43600000000000005
Q5)
    k = -2 Ein = 0.02857142857142858 Eout = 0.08399999999999996
    k = -1 Ein = 0.02857142857142858 Eout = 0.05600000000000005
    k = 0 Ein = 0.0 Eout = 0.09199999999999997
    k = 1 Ein = 0.05714285714285716 Eout = 0.124
    k = 2 Ein = 0.19999999999999996 Eout = 0.22799999999999998

Process finished with exit code 0
```

- In question two, Ein value is 0.02857 and Eout value is 0.08399. I think that the answer is A) 0.03, 0.08.
- In question three, Ein value is 0.02857 and Eout is 0.07999. So the answer is D) 0.3, 0.8
- In question four, Ein value is 0.37142 and Eout is 0.43600. So the answer is E) 0.4, 0.4
- In question five, the for loop iterates -2 to 2 so we can see all Ein and Eout values. Minimum Eout is 0.056. So the answer is D) -1
- I cannot solve question six.

Part 4

Q8)



$22+3+22=47$

Q9)

minimum

10 nodes + 36 nodes = 46 nodes

Q10)

10 a 36-a

$f(a) = 10.(a-1) + a.(36-a-1) + 36-a$

$= 10a - 10 + 36a - a^2 - a + 36 - a$

$= -a^2 + 44a + 26$

$-2a + 44 = 0$

$2a = 44$

$a = 22$

$f(22) = -(22.22) + 44.22 + 26$

$= -484 + 968 + 26$

$= 510$