10주. 신경망 학습			
학번	32183164	이름	이석현

Q1 (2점) 강의 slide 15 에 있는 example 1 을 pyrhon 코드를 작성하여 실행 결과를 보이시오. (repeat 는 10 까지 한다)

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import numpy as np

def update_w(x, w):
    v = np.dot(x,w)
    e = 1 - v
    print('e:',e)
    print('delta_w:', 0.5*e*x)
    w = w + 0.5 * e * x
    print(w)
    return w

x = np.array([0.5, 0.8, 0.2])
w = np.array([0.4, 0.7, 0.8])
print('initial w:', w)

for i in range(10):
    print('%d번째' %(i+1))
    w = update_w(x,w)
```

실행화면 캡쳐:

```
In [10]: runfile('C:/Users/이석현/Desktop/강의자료/딥러닝,클라우드/
source_code_ch09/NeuralNetwork.py', wdir='C:/Users/이석현/Desktop/강의자료/
딥러닝,클라우드/source_code_ch09')
initial w: [ 0.4  0.7  0.8]
1번째e: 0.08
delta_w: [ 0.02   0.032  0.008]
[ 0.42   0.732  0.808]
2번째e: 0.0428
delta_w: [ 0.0107   0.01712  0.00428]
[ 0.4307   0.74912  0.81228]
3번째e: 0.022898
delta_w: [ 0.0057245  0.0091592  0.0022898]
[ 0.4364245  0.7582792  0.8145698]
4번째e: 0.01225043
delta_w: [ 0.00306261  0.00490017  0.00122504]
[ 0.43948711  0.76317937  0.81579484]
5번째e: 0.00655398005
delta_w: [ 0.0016385   0.00262159  0.0006554 ]
[ 0.4411256   0.76580096  0.81645024]
6번째e: 0.00350637932675
delta_w: [ 0.00087659  0.00140255  0.00035064]
[ 0.4420022   0.76720352  0.81680088]
7번째e: 0.00187591293981
delta_w: [ 0.00046898  0.00075037  0.00018759]
[ 0.44247118  0.76795388  0.81698847]
8번째e: 0.0010036134228
delta_w: [ 0.0002509   0.00040145  0.00010036]
[ 0.44272208  0.76835533  0.81708883]
9번째e: 0.000536933181198
delta_w: [  1.34233295e-04   2.14773272e-04   5.36933181e-05]
[ 0.44285631  0.7685701   0.81714252]
10번째e: 0.000287259251941
delta_w: [  7.18148130e-05   1.14903701e-04   2.87259252e-05]
[ 0.44292813  0.768685    0.81717125]
```

Q2 (2점) 강의 slide 24 에 있는 Simple Delta rule 코드를 완성하여 실행 결과를 보이시오

Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import numpy as np

def SIGMOID(x):
    return 1/(1 + np.exp(-x))

x = np.array([0.5,0.8,0.2])
w = np.array([0.4, 0.7, 0.8])
d = 1
alpha = 0.5

for i in range(50):
    v = np.sum(w*x)
    y = SIGMOID(v)
    e = d - y
    print('error', i, e)
    w = w + alpha*y*(1-y)*e*x
```

실행화면 캡쳐:

```
In [12]: runfile('C:/Users/이석현/Desktop/강의자료/딥러닝,클라우드/
source_code_ch09/NeuralNetwork.py', wdir='C:/Users/이석현/Desktop/강의자료/
딥러닝,클라우드/source_code_ch09')
error 0 0.284957894299
error 1 0.279488769193
error 2 0.274249101076
error 3 0.269226147839
error 4 0.264407920634
error 5 0.259783152191
error 6 0.255341262525
error 7 0.251072323273
error 8 0.246967021588
error 9 0.2430166243
error 10 0.239212942837
error 11 0.235548299287
error 12 0.23201549382
error 13 0.228607773663
error 14 0.225318803679
error 15 0.222142638612
error 16 0.219073696996
error 17 0.216106736681
error 18 0.213236831955
error 19 0.210459352171
error 20 0.207769941841
error 21 0.205164502081
error 22 0.202639173369
error 23 0.200190319512
error 24 0.197814512746
error 25 0.195508519907
error 26 0.193269289586
error 27 0.191093940205
error 28 0.188979748953
error 29 0.186924141502
error 30 0.184924682475
error 31 0.18297906658
error 32 0.18108511038
error 33 0.179240744645
error 34 0.17744400724
error 35 0.17569303652
error 36 0.173986065172
error 37 0.172321414499
error 38 0.170697489085
error 39 0.169112771834
error 40 0.167565819342
error 41 0.166055257582
error 42 0.164579777882
error 43 0.163138133165
error 44 0.16172913444
error 45 0.160351647527
error 46 0.15900458999
error 47 0.157686928267
error 48 0.156397674992
error 49 0.155135886478
```

Q3 (3점) 강의 slide 39~42 에 있는 코드를 완성하여 실행 결과를 보이시오
(실행 결과가 길므로 처음 10개와 끝 10 개 정도를 보인다)

Source code :

```
from sklearn import datasets
import random
import numpy as np

def SIGMOID(x):
    return 1/(1 + np.exp(-x))

def SLP_SGD(tr_X, tr_y, alpha, rep):
    n = tr_X.shape[1] * tr_y.shape[1]
    random.seed = 123
    w = random.sample(range(1,100), n)
    w = (np.array(w) - 50)/100
    w = w.reshape(tr_X.shape[1], -1)
    for i in range(rep):
        for k in range(tr_X.shape[0]):
            x = tr_X[k,:]
            v = np.matmul(x,w)
            y = SIGMOID(v)
            e = tr_y[k,:] - y
            temp = np.transpose(np.mat(x))*np.mat(e)
            w = w + alpha*y*(1-y)*np.array(temp)
        print('error',i,np.mean(e))
    return w

iris = datasets.load_iris()
X = iris.data
target = iris.target
num = np.unique(target, axis=0)
num = num.shape[0]
y = np.eye(num)[target]

##Train
W = SLP_SGD(X, y, alpha=0.01, rep = 1000)

##Test
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)
    pred[i] = np.argmax(y)
    print("target, predict", target[i], pred[i])
print("accuracy:", np.mean(pred==target))
```

실행화면 캡쳐:

```
In [58]: runfile('C:/Users/이석현/Desktop/강의자료/딥러닝,클라우드/
source_code_ch09/NeuralNetwork.py', wdir='C:/Users/이석현/Desktop/강의자료/
딥러닝,클라우드/source_code_ch09')
error 0 0.0276105985531
error 1 -0.0144987210928
error 2 -0.0214279379426
error 3 -0.0210121771796
error 4 -0.0201075744656
error 5 -0.0191247894303
error 6 -0.0181234591198
error 7 -0.0171321192442
error 8 -0.0161678849076
error 9 -0.0152409012295
error 10 -0.0143567320267
error 990 0.000261061923204
error 991 0.000252400717448
error 992 0.000243742627626
error 993 0.000235087660196
error 994 0.000226435821564
error 995 0.000217787118088
error 996 0.000209141556077
error 997 0.000200499141788
error 998 0.000191859881432
error 999 0.000183223781169
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 0 0.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
target, predict 2 2.0
accuracy: 0.913333333333
```

Q4 (2점) (slide 43) Practice 1 에서 α 값을 0.05, 0.1, 0.5 로 하여 테스트 하여 보시오
- 에러가 줄어드는 추세를 비교하여 보시오
- 최종 예측 accuracy 가 어떻게 되는지 비교하여 보시오

Source code :

```
from sklearn import datasets
import random
import numpy as np

def SIGMOID(x):
    return 1/(1 + np.exp(-x))

def SLP_SGD(tr_X, tr_y, alpha, rep):
    n = tr_X.shape[1] * tr_y.shape[1]
    random.seed = 123
    w = random.sample(range(1,100), n)
    w = (np.array(w) - 50)/100
    w = w.reshape(tr_X.shape[1], -1)
    for i in range(rep):
        for k in range(tr_X.shape[0]):
            x = tr_X[k,:]
            v = np.matmul(x,w)
            y = SIGMOID(v)
            e = tr_y[k,:] - y
            temp = np.transpose(np.mat(x))*np.mat(e)
            w = w + alpha*y*(1-y)*np.array(temp)
        print('error',i,np.mean(e))
    return w

iris = datasets.load_iris()
X = iris.data
target = iris.target
num = np.unique(target, axis=0)
num = num.shape[0]
y = np.eye(num)[target]

##########α=0.05############
##Train
W = SLP_SGD(X, y, alpha=0.05, rep = 1000)
##Test
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)
    pred[i] = np.argmax(y)
    print("target, predict", target[i], pred[i])
print("accuracy:", np.mean(pred==target))
```

```
##########α=0.1############
##Train
W = SLP_SGD(X, y, alpha=0.1, rep = 1000)
##Test
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)
    pred[i] = np.argmax(y)
    print("target, predict", target[i], pred[i])
print("accuracy:", np.mean(pred==target))

##########α=0.5############
##Train
W = SLP_SGD(X, y, alpha=0.5, rep = 1000)
##Test
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)
    pred[i] = np.argmax(y)
    print("target, predict", target[i], pred[i])
print("accuracy:", np.mean(pred==target))
```

실행화면 캡쳐:

```
alpha = 0.05
error 0 -0.00471313917994
error 1 -0.00592568317957
error 2 -0.00512165577074
error 3 -0.00416816757731
error 4 -0.00330929820423
error 5 -0.00259115849911
error 6 -0.00199640364462
error 7 -0.00149762088087
error 8 -0.00107231415531
error 9 -0.000704354886623
error 10 -0.000382501663025
error 11 -9.87978595216e-05
error 12 0.000152599576011
error 13 0.000376177064743
error 14 0.000575526080776
error 15 0.000753618209634
error 16 0.000912970137384
error 17 0.00105574620154
error 18 0.00118382596096
error 19 0.00129885188107
error 20 0.00140226522934
accuracy: 0.88
```

```
alpha = 0.1
error 0 -0.00206341978687
error 1 -0.00256043061748
error 2 -0.00187584813057
error 3 -0.00122763109774
error 4 -0.00073493603071
error 5 -0.000373992030656
error 6 -9.07872443472e-05
error 7 0.000135270659376
error 8 0.000318050130594
error 9 0.000469153901378
error 10 0.000596231845546
error 11 0.000703811182955
error 12 0.000794670937593
error 13 0.000870766680232
error 14 0.00093369271685
error 15 0.000984873389663
error 16 0.00102562814966
error 17 0.00105718701453
error 18 0.0010806907315
error 19 0.0010971889753
error 20 0.00110764063703
accuracy: 0.873333333333
```

```
alpha = 0.5
error 0 -0.00619854964073
error 1 0.00182303386842
error 2 -0.00686009869195
error 3 -0.00254457471164
error 4 -0.00160763872788
error 5 -0.00146657249716
error 6 0.00407858201998
error 7 0.000138102904394
error 8 0.00310473018646
error 9 -3.74734679256e-05
error 10 -0.00198016836394
error 11 -0.00199614364238
error 12 -0.00144489133197
error 13 -0.00197947891396
error 14 -0.00205168550584
error 15 -0.00158487825196
error 16 -0.00143586925645
error 17 -0.00133129601597
error 18 -0.00146678864193
error 19 -0.00101728245195
error 20 -0.00166467974706
accuracy: 0.74
```

=> α값이 클수록 error가 더 빨리 줄어드는 것 같지만 최적값을 지나치게 되어 결과적으로 낮은 accuracy를 갖게 된다.

Q5 (2점) (slide 43) Practice 1에서 α 값은 0.01 로 하고 repeat time 을 200, 400, 600 으로 하여 테스트 하여 보시오
- 최종 예측 accuracy 가 어떻게 되는지 비교하여 보시오

Source code :

```
from sklearn import datasets
import random
import numpy as np

def SIGMOID(x):
    return 1/(1 + np.exp(-x))

def SLP_SGD(tr_X, tr_y, alpha, rep):
    n = tr_X.shape[1] * tr_y.shape[1]
    random.seed = 123
    w = random.sample(range(1,100), n)
    w = (np.array(w) - 50)/100
    w = w.reshape(tr_X.shape[1], -1)
    for i in range(rep):
        for k in range(tr_X.shape[0]):
            x = tr_X[k,:]
            v = np.matmul(x,w)
            y = SIGMOID(v)
            e = tr_y[k,:] - y
            temp = np.transpose(np.mat(x))*np.mat(e)
            w = w + alpha*y*(1-y)*np.array(temp)
        print('error',i,np.mean(e))
    return w

iris = datasets.load_iris()
X = iris.data
target = iris.target

num = np.unique(target, axis=0)
num = num.shape[0]
y = np.eye(num)[target]
```

```
##########repeat=200############
##Train
W = SLP_SGD(X, y, alpha=0.01, rep = 200)

##Test
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)

    pred[i] = np.argmax(y)
    print("target, predict", target[i], pred[i])
print("accuracy:", np.mean(pred==target))

##########repeat=400############
##Train
W = SLP_SGD(X, y, alpha=0.01, rep = 400)

##Test
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)

    pred[i] = np.argmax(y)
    print("target, predict", target[i], pred[i])
print("accuracy:", np.mean(pred==target))

##########repeat=600############
##Train
W = SLP_SGD(X, y, alpha=0.01, rep = 600)

##Test
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)

    pred[i] = np.argmax(y)
    print("target, predict", target[i], pred[i])
print("accuracy:", np.mean(pred==target))
```

실행화면 캡쳐:

```
repeat time = 200
accuracy: 0.8
repeat time = 400
accuracy: 0.853333333333
repeat time = 600
accuracy: 0.886666666667
```

Q6 (4점) Practice 1을 수정하되 학습률 α=0.01, epoch= 50, batch size=10 으로 하고 dataset을 train/test 로 나누되 test의 비율은 30%로 하시오.
- training accuracy 와 test accuracy를 보이시오

Source code :

```
from sklearn import datasets
import random
import numpy as np
from sklearn.model_selection import train_test_split

def SIGMOID(x):
    return 1/(1 + np.exp(-x))

def getMatrixMean(matrix):
    sum = [[0]*matrix.shape[2] ]*matrix.shape[1]
    for k in range(matrix.shape[0]):
        for i in range(matrix.shape[1]):
            for j in range(matrix.shape[2]):
                sum[i][j] += matrix[k][i][j]
    return sum

def mini_batch(tr_X, tr_y, alpha, epoch, batch_size):
    n = tr_X.shape[1] * tr_y.shape[1]
    random.seed = 123
    w = random.sample(range(1,100), n)
    w = (np.array(w) - 50)/100
    w = w.reshape(tr_X.shape[1], -1)
    for i in range(epoch):
        w_updated = []
        for k in range(tr_X.shape[0]):
            x = tr_X[k,:]
            v = np.matmul(x,w)
            y = SIGMOID(v)
            #print('y',y)
            e = tr_y[k,:] - y
            temp = np.transpose(np.mat(x))*np.mat(e)
            #print('temp',temp)
            w_updated.append(w + alpha*y*(1-y)*np.array(temp))
        w = getMatrixMean(np.array(w_updated))
        print('error',i,np.mean(e))
    return w
```

```
def getTarget(test_y):
    target = np.zeros(test_y.shape[0])
    for i in range(len(test_y)):
        for j in range(test_y.shape[1]):
            if test_y[i][j] == 1:
                target[i] = j
    return target

iris = datasets.load_iris()
X = iris.data
#type(iris.target)
target = iris.target
#print(target)
num = np.unique(target, axis=0)
num = num.shape[0]
y = np.eye(num)[target]

train_X, test_X, train_y, test_y = \
    train_test_split(X, y, test_size=0.3,\
                     random_state=1234)
##Train
W = mini_batch(train_X, train_y, alpha=0.01, epoch = 5, batch_size =
10)
pred = np.zeros(X.shape[0])
for i in range(X.shape[0]):
    v = np.matmul(X[i,:],W)
    y = SIGMOID(v)

    pred[i] = np.argmax(y)
    #print("target, predict", target[i], pred[i])
print("training accuracy:", np.mean(pred==target))
##Test
test_target = getTarget(test_y)
pred = np.zeros(test_X.shape[0])
for i in range(test_X.shape[0]):
    v = np.matmul(test_X[i,:],W)
    y = SIGMOID(v)

    pred[i] = np.argmax(test_y)
    #print("target, predict", test_target[i], pred[i])
print("test accuracy:", np.mean(pred==test_target))
```

실행화면 캡쳐:

```
In [62]: runfile('C:/Users/이석현/Desktop/강의자료/딥러닝,클라우드/
source_code_ch09/NeuralNetwork.py', wdir='C:/Users/이석현/Desktop/강의자료/
딥러닝,클라우드/source_code_ch09')
error 0 -0.0900238235067
error 1 0.0
error 2 0.0
error 3 0.0
error 4 0.0
training accuracy: 0.333333333333
test accuracy: 0.377777777778
```

코드의 어느 부분에서 문제가 있어 정확도가 낮게 형성되는지 이유를 못 찾겠습니다.