10주. 신경망 학습			
학번	32153180	이름	이상민

```
# library and function

from sklearn import datasets
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
import numpy as np
import random

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 = 100
    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
            w = w + alpha * np.matmul(x.reshape(-1, 1), \
                                      (y * (1-y) * e).reshape(1, -1))
        print("error", i, np.mean(e))
    return w
```

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

Source code :

```
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(10):
    v = np.sum(w * x)
    y = v
    e = d - y
    delta_w = alpha * e * x
    w += delta_w

    print('\n<< repeat', i+1, '>>')
    print('   e =', e)
    print('   delta w =', delta_w)
    print('   w =', w, end='\n')
```

실행화면 캡쳐:

```
<< repeat 1 >>
   e = 0.07999999999999996
   delta w = [0.02  0.032 0.008]
   w = [0.42  0.732 0.808]

<< repeat 2 >>
   e = 0.04279999999999995
   delta w = [0.0107  0.01712 0.00428]
   w = [0.4307  0.74912 0.81228]

<< repeat 3 >>
   e = 0.022897999999999863
   delta w = [0.0057245 0.0091592 0.0022898]
   w = [0.4364245 0.7582792 0.8145698]

<< repeat 4 >>
   e = 0.012250430000000034
   delta w = [0.00306261 0.00490017 0.00122504]
   w = [0.43948711 0.76317937 0.81579484]
```

```
<< repeat 5 >>
   e = 0.006553980049999963
   delta w = [0.0016385  0.00262159 0.0006554 ]
   w = [0.4411256  0.76580096 0.81645024]

<< repeat 6 >>
   e = 0.00350637932675002
   delta w = [0.00087659 0.00140255 0.00035064]
   w = [0.4420022  0.76720352 0.81680088]

<< repeat 7 >>
   e = 0.001875912939811153
   delta w = [0.00046898 0.00075037 0.00018759]
   w = [0.44247118 0.76795388 0.81698847]

<< repeat 8 >>
   e = 0.0010036134227988658
   delta w = [0.0002509  0.00040145 0.00010036]
   w = [0.44272208 0.76835533 0.81708883]

<< repeat 9 >>
   e = 0.0005369331811975186
   delta w = [1.34233295e-04 2.14773272e-04 5.36933181e-05]
   w = [0.44285631 0.7685701  0.81714252]

<< repeat 10 >>
   e = 0.0002872592519406192
   delta w = [7.18148130e-05 1.14903701e-04 2.87259252e-05]
   w = [0.44292813 0.768685   0.81717125]
```

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

Source code :

```
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 + (a * y * (1-y) * e * x)
```

실행화면 캡쳐:

```
error 0 0.2849578942990102
error 1 0.2794887691927339
error 2 0.2742491010755598
error 3 0.26922614783872123
error 4 0.26440792063416385
error 5 0.25978315219123826
error 6 0.25534126252533806
error 7 0.25107232327280227
error 8 0.2469670215879135
error 9 0.24301662429965365
error 10 0.23921294283737404
error 11 0.23554829928650334
error 12 0.23201549382012487
error 13 0.22860777366327356
error 14 0.22531880367881096
error 15 0.222142638612413
error 16 0.21907369699602874
error 17 0.2161067366812902
error 18 0.21323683195453502
error 19 0.21045935217148426
error 20 0.20776994184079545
error 21 0.20516450208052606
error 22 0.20263917336909443
error 23 0.20019031951192723
error 24 0.19781451274606
error 25 0.1955085199071097
error 26 0.19326928958591072
error 27 0.19109394020546977
error 28 0.188979748952539
error 29 0.18692414150189518
```

```
error 31 0.1829790665804365
error 32 0.18108511038041042
error 33 0.17924074464474082
error 34 0.17744400724015885
error 35 0.17569303651953594
error 36 0.17398606517187687
error 37 0.1723214144986963
error 38 0.1706974890847862
error 39 0.1691127718338511
error 40 0.16756581934176817
error 41 0.1660552575823464
error 42 0.16457977788241818
error 43 0.16313813316490478
error 44 0.16172913444016468
error 45 0.16035164752747189
error 46 0.15900458998988964
error 47 0.15768692826710384
error 48 0.15639767499198376
error 49 0.15513588647773924
```

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 :

```
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]

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

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 :', round(np.mean(pred==target), 4))
```

실행화면 캡쳐:

```
error 0 -0.005941909737521771
error 1 -0.019684720507359883
error 2 -0.022748348633018334
error 3 -0.023973546444212717
error 4 -0.023831540741735397
error 5 -0.022983726036986545
error 6 -0.02184682238587014
error 7 -0.02061250699875707
error 8 -0.019367297423327916
error 9 -0.018152234312429554
error 10 -0.01698749055515199
```

```
error 990 -8.9677087367665e-05
error 991 -9.772585535019566e-05
error 992 -0.0001057727067789449
error 993 -0.00011381763198030821
error 994 -0.00012186062134325604
error 995 -0.00012990166531913946
error 996 -0.00013794075441998324
error 997 -0.00014597787921851826
error 998 -0.00015401303034882907
error 999 -0.00016204619850391141
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
accuracy : 0.9133
```

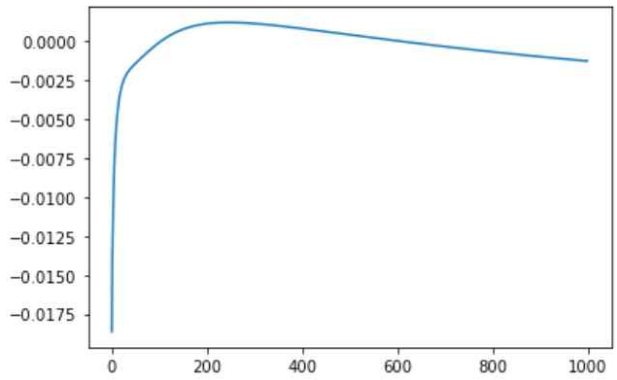
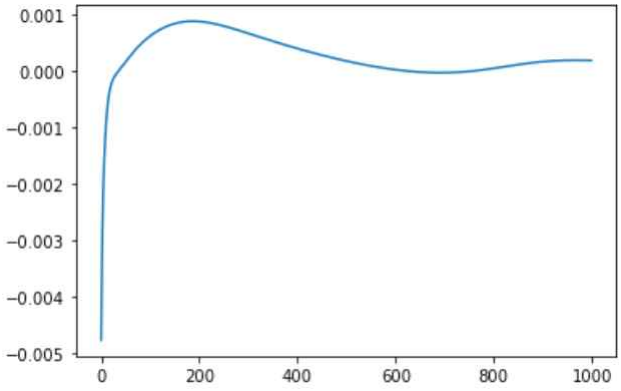
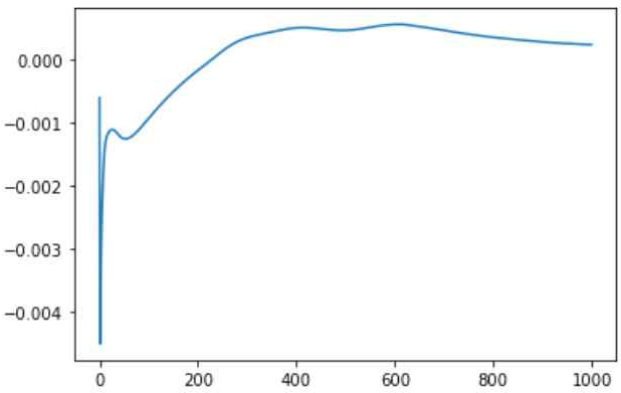
Q4 (2점) (slide 43) Practice 1 에서 α 값을 0.05, 0.1, 0.5 로 하여 테스트 하여 보시오
- 에러가 줄어드는 추세를 비교하여 보시오
- 최종 예측 accuracy 가 어떻게 되는지 비교하여 보시오

Source code :

```
def SLP_SGD(tr_X, tr_y, alpha, rep):
    n = tr_X.shape[1] * tr_y.shape[1]
    random.seed = 100
    w = random.sample(range(1, 100), n)
    w = (np.array(w) - 50) / 100
    w = w.reshape(tr_X.shape[1], -1)

    coord_y = []

    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
            w = w + alpha * np.matmul(x.reshape(-1, 1), \
                                      (y * (1-y) * e).reshape(1, -1))
        coord_y.append(np.mean(e))
        print("error", i, np.mean(e))
    plt.plot(np.array(range(1000)), coord_y)
    plt.show()
    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]

alpha = [0.05, 0.1, 0.15]
W = [SLP_SGD(X, y, alpha=alpha[0], rep=1000),
     SLP_SGD(X, y, alpha=alpha[1], rep=1000),
     SLP_SGD(X, y, alpha=alpha[2], rep=1000)]

pred = [np.zeros(X.shape[0]),
        np.zeros(X.shape[0]),
        np.zeros(X.shape[0])]

for i in range(3):
    for j in range(X.shape[0]):
        v = np.matmul(X[j,:], W[i])
        y = SIGMOID(v)

        pred[i][j] = np.argmax(y)

for i in range(3):
    print('alpha {}, accuracy: {}'.format(alpha[i],
                                          round(np.mean(pred[i]==target),
4)))
```

실행화면 캡쳐:

alpha = 0.05	
alpha = 0.1	
alpha = 0.15	

```
alpha 0.05, accuracy: 0.88
alpha 0.1, accuracy: 0.8733
alpha 0.15, accuracy: 0.86
```

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 :

```
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, 400, 600]
W = [SLP_SGD(X, y, alpha=0.01, rep=repeat[0]),
     SLP_SGD(X, y, alpha=0.01, rep=repeat[1]),
     SLP_SGD(X, y, alpha=0.01, rep=repeat[2])]

pred = [np.zeros(X.shape[0]),
        np.zeros(X.shape[0]),
        np.zeros(X.shape[0])]

for i in range(3):
    for j in range(X.shape[0]):
        v = np.matmul(X[j,:], W[i])
        y = SIGMOID(v)

        pred[i][j] = np.argmax(y)

for i in range(3):
    print('repeat {}, accuracy: {}'.format(repeat[i],
                                           round(np.mean(pred[i]==target),
4)))
```

실행화면 캡쳐:

```
repeat 200, accuracy: 0.8
repeat 400, accuracy: 0.86
repeat 600, accuracy: 0.8867
```

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 :

```
def SLP_SGD_EPOCH(tr_X, tr_y, alpha, epoch, batch):
    n = tr_X.shape[1] * tr_y.shape[1]
    random.seed = 100
    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):
        for j in range(len(batch)):
            first = batch[j][0]
            last = batch[j][1]

            delta_w = [[0, 0, 0]] * 4
            for k in range(first, last, 1):
                x = tr_X[k, :]
                v = np.matmul(x, w)
                y = SIGMOID(v)
                e = tr_y[k, :] - y
                delta_w += np.matmul(x.reshape(-1, 1), \
                                     (alpha * y * (1-y) * e).reshape(1,
-1))
            delta_w = delta_w / (last - first)
            w += delta_w
    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_X, test_X, train_target, test_target = \
    train_test_split(X, target, test_size=0.3, random_state=111)

num = np.unique(train_target, axis=0)
num = num.shape[0]
train_y = np.eye(num)[train_target]

num = np.unique(test_target, axis=0)
num = num.shape[0]
test_y = np.eye(num)[test_target]

batch = [[0, 10], [10, 20], [20, 30], [30, 40], [40, 50],
       [50, 60], [60, 70], [70, 80], [80, 90], [90, 100], [100, 105]]

W = SLP_SGD_EPOCH(train_X, train_y, alpha=0.01, epoch=50,
batch=batch)

train_pred = np.zeros(train_X.shape[0])
test_pred = np.zeros(test_X.shape[0])

for i in range(train_X.shape[0]):
    v = np.matmul(train_X[i, :], W)
    train_y = SIGMOID(v)
    train_pred[i] = np.argmax(train_y)

for i in range(test_X.shape[0]):
    v = np.matmul(test_X[i, :], W)
    test_y = SIGMOID(v)
    test_pred[i] = np.argmax(test_y)

print('train accuracy: ', round(np.mean(train_pred==train_target),
4))
print('test accuracy: ', round(np.mean(test_pred==test_target), 4))
```

실행화면 캡쳐:

```
train accuracy:  0.6667
test accuracy:  0.6889
```