Deep Learning 실습 중간고사 (10점)

학과 : **학번 :** **이름**

1. 다음은 당뇨병 데이터를 받아와 'diabetes_X'의 3번째 열만을 선택하여 5-fold cross validation을 하기 위해 데이터를 분할하는 코드이다. 빈칸을 채워 완성시키시오. (3점)

Python

```
import matplotlib.pylab as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
diabetes_X_new = diabetes_X[:, np.newaxis, [ 2 ] ]  # 3번째 열의 BMI 특징만 선택

X_train, X_test, y_train, y_test = [ train_test_split ] ( [ diabetes_X_new ] ,
[ diabetes_y ] , [ test_size=0.2 ] )
```

2. 다음은 단층 퍼셉트론으로 **그림 1**의 논리 게이트를 학습시키는 코드이다. 빈칸을 채워 다음 코드를 완성시키시오. (3점)

Python

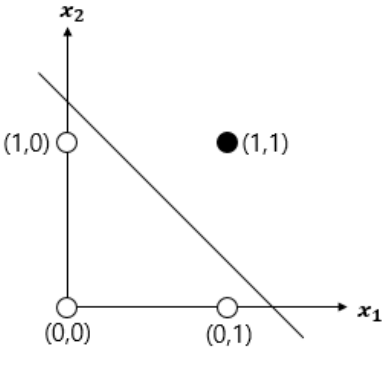
```
import numpy as np
epsilon = 0.0000001
def step_func(t):
    if t > epsilon: return 1
    else: return 0
X = np.array([ [0, 0, 1] , [0, 1, 1] , [1, 0, 1] , [1, 1, 1] ])
y = np.array([ 0, 0, 0, 1   or 1, 1, 1, 0 ])
W = np.zeros(len(X[0]))
def perceptron_predict(X, Y):
    global W
    for x in X:
        print(x[0], x[1], "->", step_func(np.dot(x, W)))
```

```
def perceptron_fit(X, Y, epochs):
    global W
    eta = 0.2
    for t in range(epochs):
        for i in range(len(X)):
            predict = step_func(np.dot(X[i], W))
            error = Y[i] - predict
            W  W += eta * error * X[i]

W += eta * error * np.dot( X[i], W)

perceptron_fit(X, y, 6)
```

그림 1

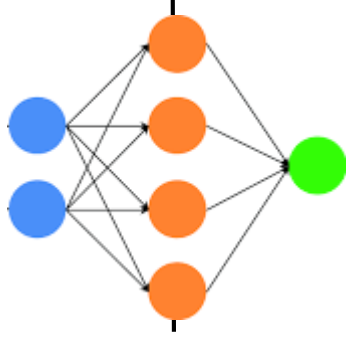


3. 다음은 **그림2**의 논리 게이트를 학습시키는 MLP 코드이다 좌측에 일반 변수로 만들어진 코드를 우측의 넘파이 코드로 바꾸고, 역전파 함수의 빈칸을 채우고, 해당 코드의 레이어 구조를 그림으로 그려라 (4점)

Python

```
# 일반 변수
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_deriv(x):
    return x * (1 - x)
learning_rate = 0.2
inputs = [[0, 0] , [0, 1] , [1, 0] , [1, 1]]
targets = [ 0, 1, 1, 0 or 1, 0, 0, 1 ]
w1_11, w1_21, b1_1 = 0.10, 0.30, 0.1
w1_12, w1_22, b1_2 = 0.20, 0.40, 0.2
w1_13, w1_23, b1_3 = 0.15, 0.35, 0.15
w1_14, w1_24, b1_4 = 0.25, 0.45, 0.05

w2_11, w2_21, w2_31, w2_41 = 0.50, 0.60, 0.55, 0.65
b2 = 0.3
def forward(x):
    global w1_11, w1_21, b1_1, w1_12, w1_22, b1_2, w1_13,
 w1_23, b1_3, w1_14, w1_24, b1_4
    global w2_11, w2_21, w2_31, w2_41, b2
    x1, x2 = x
    z1 = x1 * w1_11 + x2 * w1_21 + b1_1
    a1 = sigmoid(z1)
    z2 = x1 * w1_12 + x2 * w1_22 + b1_2
    a2 = sigmoid(z2)
    z3 = x1 * w1_13 + x2 * w1_23 + b1_3
    a3 = sigmoid(z3)
    z4 = x1 * w1_14 + x2 * w1_24 + b1_4
    a4 = sigmoid(z4)
    z_out = a1 * w2_11 + a2 * w2_21 + a3 * w2_31 +
a4 * w2_41 + b2
    a_out = sigmoid(z_out)
    return a1, a2, a3, a4, a_out
```



```
#np.array 사용
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_deriv(x):
    return x * (1 - x)
learning_rate = 0.2
X = np.array([[0, 0] , [0, 1] , [1, 0] , [1, 1]])
T = np.array( [ [1], [0], [0], [1] ])
learning_rate = 0.2
W1 = np.array( [ [0.10, 0.20, 0.15, 0.25], [0.30, 0.40, 0.35, 0.45] ] )
B1 = np.array( [0.1, 0.2, 0.15, 0.05] )
W2= np.array( [ [0.50], [0.60], [0.55], [0.65] ] )
B2= np.array( [0.3] )
def forward(x):
    z1 = np.dot(x, W1) + B1
    a1 = sigmoid(z1)
    z2 = np.dot(a1, W2) + B2
    a2 = sigmoid(z2)
    return x, a1, a2
def train(epochs=10000):
    global W1, W2, B1, B2
    for _ in range(epochs):
        for x, t in zip(X, T):
            x = x.reshape(1, -1)
            t = t.reshape(1, -1)
            layer0, layer1, layer2 = forward(x)
            error = layer2 - t
            delta2 = error*sigmoid_deriv(layer2)
            error1 = np.doc(delta2, W2.T)
            delta1 = error1*sigmoid_deriv(layer1)
            W2 -= learning_rate * np.dot( layer1.T, delta2 )
            B2 -= learning_rate * delta2
            W1 -= learning_rate * np.dot( layer0.T, delta1 )
            B1 -= learning_rate * delta1
```

답: 1) 0, 1, 1, 0 or 1, 0, 0, 1 2) ([[1], [0], [0], [1]]) 차원달라도됨 3) ([[0.10, 0.20, 0.15, 0.25], [0.30, 0.40, 0.35, 0.45]])

4) ([0.1, 0.2, 0.15, 0.05]) 5) ([[0.50], [0.60], [0.55], [0.65]]) 6) ([0.3]) 어떤형태든 1개

7) layer2 – t 8) error* sigmoid_deriv(layer2) 9) np.dot(delta2, W2.T)

10) error1* sigmoid_deriv (layer1) 11) layer1.T, delta2 12) layer0.T, delta1

그림 2