# Metode Multilayer Perceptron

## Gelar Bayu Adirama (171011400331)

```
import numpy as np
ls = np.array([2, 4, 4, 1])
n = len(ls)
W = []
for i in range(n - 1):
    W.append(np.random.randn(ls[i], ls[i + 1]) * 0.1)
B = []
for i in range(1, n):
    B.append(np.random.randn(ls[i]) * 0.1)
O = []
for i in range(n):
    O.append(np.zeros([ls[i]]))
D = []
for i in range(1, n):
    D.append(np.zeros(ls[i]))
A = np.matrix([[0.0, 0.0], [0.0, 1.0], [1.0, 0.0], [1.0, 1.0]])
#Target Vectors (1 row per each)
y = np.matrix([[-0.5], [0.5], [0.5], [-0.5]])
actF = []
dF = []
for i in range(n - 1):
    actF.append(lambda x : np.tanh(x))
    dF.append(lambda y : 1 - np.square(y))
 actF.append(lambda x: x)
```

```
dF.append(lambda x : np.ones(x.shape))
 a = 0.5
numIter = 250
for c in range(numIter):
    for i in range(len(A)):
        print(str(i))
        t = y[i, :]
        O[0] = A[i, :]
        for j in range(n - 1):
            O[j + 1] = actF[j](np.dot(O[j], W[j]) + B[j])
        print('Out:' + str(O[-1]))
        D[-1] = np.multiply((t - O[-1]), dF[-1](O[-1]))
        for j in range(n - 2, 0, -1):
            D[j - 1] = np.multiply(np.dot(D[j], W[j].T), dF[j](O[j]))
        for j in range(n - 1):
            W[j] = W[j] + a * np.outer(O[j], D[j])
            B[j] = B[j] + a * D[j]
 print('\nFinal weights:')
for i in range(n - 1):
    print('Layer ' + str(i + 1) + ':\n' + str(W[i]) + '\n')
```

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\bayu\.spyder-py3\autosave

C:\Users\bayu\.spyder-py3\autosave\UAS_MultilayerPerceptron_171011400331_GelarBayuAdirama.py

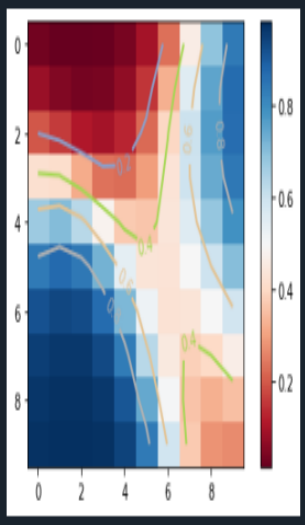
temp.py | UAS_MultilayerPerceptron_171011400331_GelarBayuAdirama.py

```
    O.append(np.zeros([ls[i]]))
D = []
for i in range(1, n):
    D.append(np.zeros(ls[i]))
A = np.matrix([[0.0, 0.0], [0.0, 1.0], [1.0, 0.0], [1.0, 1.0]])
#Target Vectors (1 row per each)
y = np.matrix([[-0.5], [0.5], [0.5], [-0.5]])
actF = []
dF = []
for i in range(n - 1):
    actF.append(lambda x : np.tanh(x))
    dF.append(lambda y : 1 - np.square(y))

actF.append(lambda x: x)
dF.append(lambda x : np.ones(x.shape))

a = 0.5
numIter = 250
for c in range(numIter):
    for i in range(len(A)):
        print(str(i))
        t = y[i, :]
        O[0] = A[i, :]
        for j in range(n - 1):
            O[j + 1] = actF[j](np.dot(O[j], W[j]) + B[j])
        print('Out:' + str(O[-1]))
        D[-1] = np.multiply((t - O[-1]), dF[-1](O[-1]))
        for j in range(n - 2, 0, -1):
            D[j - 1] = np.multiply(np.dot(D[j], W[j].T), dF[j](O[j]))
        for j in range(n - 1):
            W[j] = W[j] + a * np.outer(O[j], D[j])
            B[j] = B[j] + a * D[j]

print('\nFinal weights:')
for i in range(n - 1):
    print('Layer ' + str(i + 1) + ':\n' + str(W[i]) + '\n')
```




Variable explorer | Plots | Files

Console 1/A

```
1
Out:[[0.253958397106844]]
2
Out:[[-0.369531330476879]]
3
Out:[[0.283098028758226]]
0
Out:[[-0.181703637766623]]
1
Out:[[0.261635809347732]]
2
Out:[[-0.361799221212288]]
3
Out:[[0.273233037463693]]
0
Out:[[-0.190405969759486]]
```

IPython console | History

LSP Python: ready | conda: base (Python 3.8.3) | Line 35, Col 23 | UTF-8 | CRLF | RW | Mem 63%

Type here to search

ENG 9:29 PM 11/15/2020