Latihan UTS PBF

May 16, 2022

1 Latihan UAS

Buatlah fungsi untuk membangkitkan matriks w, m, h secara random dengan batas bawah -1 dan batas atas 1.

[6]:
```
import numpy as np

gen_rand = lambda a, b, c: np.random.uniform(low = a, high = b, size = c)
W = gen_rand(-1, 1, (3,2))
M = gen_rand(-1, 1, (2,2))
H = gen_rand(-1, 1, (1,1))
```

Fungsi aktivasi untuk forward propagation

[4]:
```
# code
import math
import functools as ft

''' Fungsi aktivasi '''
def aktivasi(x):
    return (math.exp(x)-math.exp(-x))/(math.exp(x)+math.exp(-x))

'''
Mendapatkan nilai W di setiap list sesuai dengan index.
Mapping W menjadi satu dimensi
'''
def WTi(W, i):
    return list(map( lambda w:w[i], W))

'''
Menampung WTi sesuai index dan membuat menjadi satu list
'''
def WT(W):
    return list( map( lambda i : WTi(W, i), range(len(W[0])) ) )

'''
Nilai yang masuk ke neuron di hidden layer adalah penjumlahan antara perkalian␣
 ↪weight dengan
```

```
nilai yang masuk pada input neuron.
'''
def XW(X,W):
    return map( lambda w: ft.reduce( lambda a,b:a+b, map( lambda xx,vw: xx*vw,␣
 ↪X, w), 0), WT(W) )

''' Mengaktivasi nilai yang didapat pada XW '''
def input_to_hidden(X, W):
    return list( map( lambda x:aktivasi(x) , XW(X, W) ) )

''' membuat feed-forward dari fungsi yang sudah dibuat di atas, supaya modular␣
 ↪'''
@calc_time_decorator
def feed_forward(X, W, M, H):
    return input_to_hidden(input_to_hidden(input_to_hidden(X, W), M), H)
```

[5]:
```
X = [ 0, 8, 1]
forw_prop = feed_forward(X, W, M, H)
print(f'Output: {forw_prop}')
```

```
Time:  0.00011444091796875
Output: [-0.31466743954210885]
```

Buatlah decorator untuk menambahkan perhitungan running time

[3]:
```
import time

def calc_time_decorator(func):
    def inner(*args, **kwargs):
        start = time.time()
        res = func(*args, **kwargs)
        end = time.time()
        print('Time: ', end - start)
        return res
    return inner
```

[ ]:
```
@calc_time_decorator
```

2 Jurnal - Abaikan saja

[ ]:
```
import math
import functools as ft

''' Fungsi aktivasi '''
def aktivasi(x):
    return 1/ (1+ math.exp(-x))
```

```
'''
Mendapatkan nilai W di setiap list sesuai dengan index.
Mapping W menjadi satu dimensi
'''
def WTi(W, i):
    return list(map( lambda w:w[i], W))

'''
Menampung WTi sesuai index dan membuat menjadi satu list
'''
def WT(W):
    return list( map( lambda i : WTi(W, i), range(len(W[0])) ) )

'''
Nilai yang masuk ke neuron di hidden layer adalah penjumlahan antara perkalian␣
 ↪weight dengan
nilai yang masuk pada input neuron.
'''
def XW(X,W):
    return map( lambda w: ft.reduce( lambda a,b:a+b, map( lambda xx,vw: xx*vw,␣
 ↪X, w), 0), WT(W) )

''' Mengaktivasi nilai yang didapat pada XW '''
def input_to_hidden(X, W):
    return list( map( lambda x:aktivasi(x) , XW(X, W) ) )

''' membuat feed-forward dari fungsi yang sudah dibuat di atas, supaya modular␣
 ↪'''
def feed_forward(X, W, M):
    return input_to_hidden(input_to_hidden(X, W), M)
```

[ ]:
```
X = [ 9, 10, -4 ]
W = [ [ 0.5, 0.4 ] , [ 0.3, 0.7 ] , [ 0.25, 0.9 ] ]
M = [ [ 0.34 ], [0.45] ]

feed_forward(X, W, M)
```

[ ]:
```
[0.6876336740661236]
```

[ ]:
```
input_to_hidden(X, W)

print('1')
print(list(XW(X,W)))
print(input_to_hidden(X,W))

print('2')
print(list(XW(input_to_hidden(X,W),W)))
```

```
print(input_to_hidden(input_to_hidden(X, W), M))
```

```
1
[6.5, 7.0]
[0.998498817743263, 0.9990889488055994]
2
[0.7989760935133112, 1.0987617912612246]
[0.6876336740661236]
```