cwiczenie1

March 28, 2023

[ ]:
```
# Adrian Kokoszka 19727
# importowanie potrzebnych bibliotek
import tensorflow as tf #biblioteka tensorflow
import numpy as np #biblioteka numpy, aby móc korzystać z funkcji związanych z␣
 ↪obliczeniami matematycznymi
import matplotlib.pyplot as plt #biblioteka matplotlib, aby móc korzystać z␣
 ↪funkcji związanych z rysowaniem wykresów
from mpl_toolkits.mplot3d import Axes3D #biblioteka do rysowania wykresów 3D
print(tf.__version__) #sprawdzenie wersji biblioteki tensorflow
```

```
2.11.0
```

[ ]:
```
observations=1000 #ilość obserwacji
xs = np.random.uniform(low=-10,high=10, size=(observations,1)) #losowanie␣
 ↪wartości z zakresu od -10 do 10 i zapisanie ich do tablicy xs
xz = np.random.uniform(low=-10,high=10, size=(observations,1)) #losowanie␣
 ↪wartości z zakresu od -10 do 10 i zapisanie ich do tablicy xz
inputs=np.column_stack((xs,xz)) #łączenie dwóch tablic w jedną
print(inputs.shape) #wypisanie rozmiaru tablicy, w tym przypadku 1000 wierszy i␣
 ↪2 kolumny
```

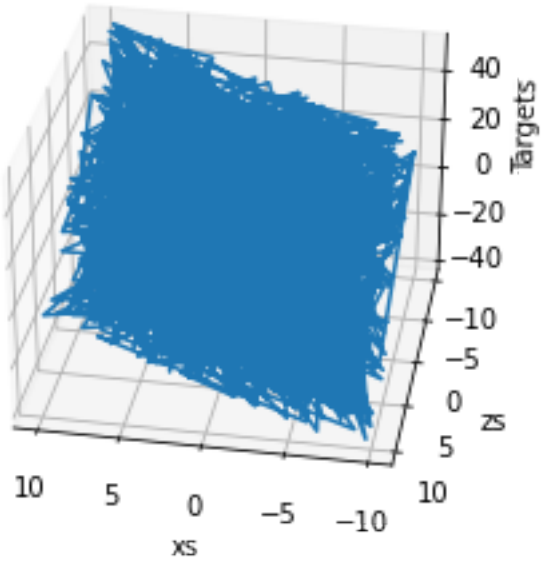
```
(1000, 2)
```

[ ]:
```
noise = np.random.uniform(low=-1,high=1, size=(observations,1)) # losowanie␣
 ↪wartości szumu z zakresu od -1 do 1 i zapisanie ich do tablicy noise
targets = 2*xs - 3*xz + 5 + noise #funkcja liniowa, która ma być wyznaczona␣
 ↪przez sieć neuronową
np.savez('TF_dataset', inputs=inputs, targets=targets)# zapisanie danych do␣
 ↪pliku .npz
print(targets.shape) # wypisanie rozmiaru tablicy, w tym przypadku 1000 wierszy␣
 ↪i 1 kolumna
```

```
(1000, 1)
```

[ ]:
```
targets = targets.reshape(observations,) #zmiana rozmiaru tablicy targets bez␣
 ↪zmiany jej zawartości
xs = xs.reshape(observations,) #zmiana rozmiaru tablicy xs, bez zmiany jej␣
 ↪zawartości
```

```
xz = xz.reshape(observations,) #zmiana rozmiaru tablicy xz bez zmiany jej
 ↪zawartości
fig = plt.figure() #tworzenie wykresu
ax = fig.add_subplot(111, projection='3d') #wyswietlanie wykresu w trzech
 ↪wymiarach
ax.plot(xs,xz,targets) #rysowanie wykresu
ax.set_xlabel('xs') #podpisanie osi x
ax.set_ylabel('zs') #podpisanie osi y
ax.set_zlabel('Targets') #podpisanie osi z
ax.view_init(azim=100) #ustawienie kąta widzenia wykresu na 100 stopni
plt.show() #wyświetlenie wykresu
```



[ ]:
```
init_range = 0.1 #zakres losowania wartości początkowych
weights = np.random.uniform(low=-init_range,high=init_range, size=(2,1))
 ↪#losowanie wagi z zakresu -0.1 do 0.1 i zapisanie do tablicy weights
biases = np.random.uniform(low=-init_range,high=init_range, size=1) #losowanie
 ↪odchylenia z zakresu -0.1 do 0.1 i zapisanie do tablicy biases
print(weights,biases)   #wypisanie wartości początkowych wag i odchylenia
```

```
[[0.07235325]
 [0.03127739]] [0.08300823]
```

[ ]:
```
targets = targets.reshape(observations,1) #Przekształcenie tablicy targets do
 ↪postaci 1000 wierszy (obserwacji) i 1 kolumny
eta = 0.02 #współczynnik uczenia
for i in range (100): #pętla wykonująca się 100 razy
```

```
   outputs = np.dot(inputs, weights) + biases #obliczenie wartości wyjściowej␣
↪sieci neuronowej
   deltas = outputs - targets #obliczenie wartości błędu sieci neuronowej

   loss = np.sum(deltas ** 2)/2/observations #obliczenie wartości funkcji␣
↪kosztu
   print(loss) #wypisanie wartości funkcji kosztu

   deltas_scaled = deltas/observations #obliczenie wartości deltas_scaled␣
↪dzieląc wartości deltas przez ilość obserwacji
   weights = weights - eta * np.dot(inputs.T, deltas_scaled) #obliczenie␣
↪nowych wartości weights, które są wykorzystywane w kolejnej iteracji pętli
   biases = biases - eta * np.sum(deltas_scaled) #obliczenie nowych wartości␣
↪odchylenia, które są wykorzystywane w kolejnej iteracji pętli
```

```
221.83755684234075
37.238681954449675
14.444718197761715
11.279338894310431
10.520774551651119
10.071592248821213
9.674554950915478
9.297476371833302
8.93584607400312
8.588591224328075
8.255085501139131
7.934777664081876
7.627144570083628
7.33168456486944
7.047915943782937
6.775376075770523
6.513620637716749
6.262222888616823
6.020772973675793
5.788877256097301
5.566157675332943
5.35225113072819
5.146808889557862
4.949496018485716
4.759990837521223
4.577984395583362
4.40317996681639
4.235292566836452
4.074048488120353
3.9191848537790634
3.7704491889884215
```

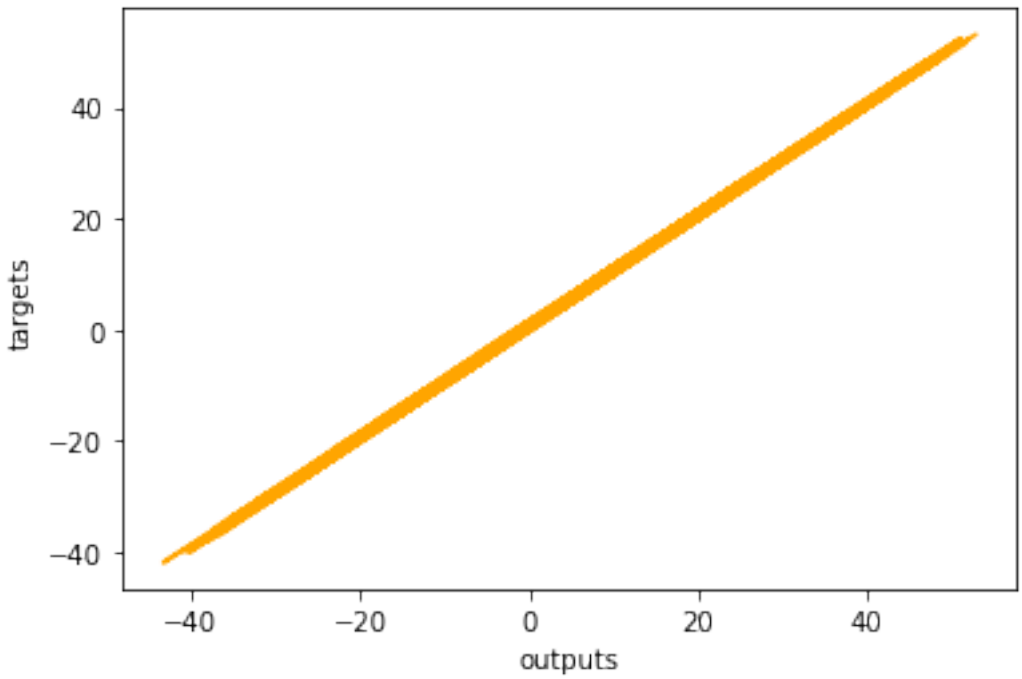
3.627599009378404
3.4904014257098357
3.3586327641940996
3.2320782018368464
3.1105314162111704
2.993794249089295
2.8816763833844017
2.7739950328759018
2.6705746442122913
2.5712466107058027
2.4758489974522218
2.3842262773277434
2.2962290774324603
2.211713935567094
2.130543066346001
2.0525841365650668
1.9777100494583388
1.9057987374916512
1.8367329633554166
1.770400128832146
1.7066920912271186
1.6455049870628955
1.586739062750262
1.5302985119595358
1.4760913194271175
1.4240291109426177
1.3740270092720257
1.3260034957819902
1.2798802775396685
1.2355821596714214
1.1930369227723086
1.1521752051664889
1.1129303898265948
1.0752384957677144
1.0390380737389324
1.004270106042354
0.9708779103163149
0.9388070471258892
0.9080052312100554
0.8784222462408146
0.8500098629553061
0.8227217605274416
0.796513451050868
0.7713422070101503
0.7471669916219296
0.7239483919324783
0.7016485545626056
0.6802311239951322

```
0.6596611833043419
0.6399051972307747
0.6209309575085494
0.6027075303560988
0.5852052060446984
0.5683954504625677
0.5522508585955948
0.5367451098488241
0.5218529251358845
0.5075500256663905
0.4938130933641272
0.48061973285149373
0.4679484349382319
0.4557785415548957
0.4440902120739253
0.4328643909633803
0.42208277672063277
0.4117277920353496
0.40178255513313244
0.39223085225309
0.38305711121447433
```

[ ]:
```
print (weights, biases) #wypisanie wartości wag i odchyleń po 100 powtórzeniach␣
↪pętli
```

```
[[ 1.9916347 ]
 [-3.00590554]] [4.34917437]
```

[ ]:
```
plt.plot(outputs, targets, color='orange') #rysowanie wykresu, gdzie osie x i y␣
↪to kolejno outputs i targets
plt.xlabel('outputs') #podpisanie osi x jako outputs
plt.ylabel('targets') #podpisanie osi y jako targets
plt.show() #wyświetlenie wykresu
```


40
20
0
−20
−40
targets
−40
−20
0
20
40
outputs

[ ]: