hw1p3

September 26, 2023

[ ]:
```
import numpy as np
from scipy.optimize import fsolve
```

Possible to have different answers for these datasets, but they will all be pretty close to making the dataset work

Dataset 1

[2]:
```
def f(p): #Formula's followed in the neural network plus the addition of␣
 ↪subtracting the outout value, as fsolve revolves around making the equation␣
 ↪equal to 0
    input = [0,1,2,3,4,5,6,7,8,9,10,11,12]
    output = [8, 6, 4, 2, 0, 2, 4, 2, 0, 2, 4, 6, 8]
    return [p[12] + p[8]*max(0,p[0]*input[0] + p[4]) + p[9]*max(0,p[1]*input[0]␣
 ↪+ p[5]) + p[10]*max(0,p[2]*input[0] + p[6]) + p[11]*max(0,p[3]*input[0] +␣
 ↪p[7]) - output[0],
    p[12] + p[8]*max(0,p[0]*input[1] + p[4]) + p[9]*max(0,p[1]*input[1] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[1] + p[6]) + p[11]*max(0,p[3]*input[1] + p[7]) -␣
 ↪output[1],
    p[12] + p[8]*max(0,p[0]*input[2] + p[4]) + p[9]*max(0,p[1]*input[2] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[2] + p[6]) + p[11]*max(0,p[3]*input[2] + p[7]) -␣
 ↪output[2],
    p[12] + p[8]*max(0,p[0]*input[3] + p[4]) + p[9]*max(0,p[1]*input[3] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[3] + p[6]) + p[11]*max(0,p[3]*input[3] + p[7]) -␣
 ↪output[3],
    p[12] + p[8]*max(0,p[0]*input[4] + p[4]) + p[9]*max(0,p[1]*input[4] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[4] + p[6]) + p[11]*max(0,p[3]*input[4] + p[7]) -␣
 ↪output[4],
    p[12] + p[8]*max(0,p[0]*input[5] + p[4]) + p[9]*max(0,p[1]*input[5] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[5] + p[6]) + p[11]*max(0,p[3]*input[5] + p[7]) -␣
 ↪output[5],
    p[12] + p[8]*max(0,p[0]*input[6] + p[4]) + p[9]*max(0,p[1]*input[6] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[6] + p[6]) + p[11]*max(0,p[3]*input[6] + p[7]) -␣
 ↪output[6],
    p[12] + p[8]*max(0,p[0]*input[7] + p[4]) + p[9]*max(0,p[1]*input[7] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[7] + p[6]) + p[11]*max(0,p[3]*input[7] + p[7]) -␣
 ↪output[7],
```

```
    p[12] + p[8]*max(0,p[0]*input[8] + p[4]) + p[9]*max(0,p[1]*input[8] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[8] + p[6]) + p[11]*max(0,p[3]*input[8] + p[7]) -␣
 ↪output[8],
    p[12] + p[8]*max(0,p[0]*input[9] + p[4]) + p[9]*max(0,p[1]*input[9] + p[5])␣
 ↪+ p[10]*max(0,p[2]*input[9] + p[6]) + p[11]*max(0,p[3]*input[9] + p[7]) -␣
 ↪output[9],
    p[12] + p[8]*max(0,p[0]*input[10] + p[4]) + p[9]*max(0,p[1]*input[10] +␣
 ↪p[5]) + p[10]*max(0,p[2]*input[10] + p[6]) + p[11]*max(0,p[3]*input[10] +␣
 ↪p[7]) - output[10],
    p[12] + p[8]*max(0,p[0]*input[11] + p[4]) + p[9]*max(0,p[1]*input[11] +␣
 ↪p[5]) + p[10]*max(0,p[2]*input[11] + p[6]) + p[11]*max(0,p[3]*input[11] +␣
 ↪p[7]) - output[11],
    p[12] + p[8]*max(0,p[0]*input[12] + p[4]) + p[9]*max(0,p[1]*input[12] +␣
 ↪p[5]) + p[10]*max(0,p[2]*input[12] + p[6]) + p[11]*max(0,p[3]*input[12] +␣
 ↪p[7]) - output[12]]
```

[3]:
```
initial_guess = np.random.normal(size =13)
solution = fsolve(f, initial_guess)
while not(np.all(np.isclose(f(solution),[0]*13)==True)):
  new_guess = np.random.normal(size =13)

  solution = fsolve(f, new_guess)
solution
```

```
C:\Users\Anil\AppData\Local\Programs\Python\Python310\lib\site-
packages\scipy\optimize\_minpack_py.py:177: RuntimeWarning: The iteration is not
making good progress, as measured by the
  improvement from the last ten iterations.
  warnings.warn(msg, RuntimeWarning)
C:\Users\Anil\AppData\Local\Programs\Python\Python310\lib\site-
packages\scipy\optimize\_minpack_py.py:177: RuntimeWarning: The iteration is not
making good progress, as measured by the
  improvement from the last five Jacobian evaluations.
  warnings.warn(msg, RuntimeWarning)
```

[3]:
```
array([-7.82455998e-01, -5.16665284e-01,  1.32120087e+00, -3.55189485e+00,
        6.25964798e+00,  3.09863545e+00, -1.05696070e+01,  1.42169031e+01,
        2.55605428e+00, -7.76233237e+00,  1.51377436e+00,  1.12912342e+00,
        2.69958647e-11])
```

[4]:
```
f(solution) #Closer to 0 the better
```

[4]:
```
[-1.751665479332587e-11,
 -2.9530156098189764e-11,
 -4.154543375989306e-11,
 -5.356071142159635e-11,
 -6.557229412229582e-11,
```

```
-1.637445734559151e-11,
-6.759481863127803e-12,
-1.2145839889399213e-12,
2.6995864717007652e-11,
3.0686564400639327e-13,
2.851052727237402e-13,
2.6290081223123707e-13,
2.398081733190338e-13]
```

Dataset 2

[5]:
```
def f(p): #Added 2 repetitive lines at the bottom of this one so that the shape␣
↪would line up but it doesn't affect the actual results
    input = [0,1,2,3,4,5,6,7,8,9,10]
    output = [0,1,2,3,2,1,2,3,4,5,6]
    return [p[12] + p[8]*max(0,p[0]*input[0] + p[4]) + p[9]*max(0,p[1]*input[0]␣
↪+ p[5]) + p[10]*max(0,p[2]*input[0] + p[6]) + p[11]*max(0,p[3]*input[0] +␣
↪p[7]) - output[0],
    p[12] + p[8]*max(0,p[0]*input[1] + p[4]) + p[9]*max(0,p[1]*input[1] + p[5])␣
↪+ p[10]*max(0,p[2]*input[1] + p[6]) + p[11]*max(0,p[3]*input[1] + p[7]) -␣
↪output[1],
    p[12] + p[8]*max(0,p[0]*input[2] + p[4]) + p[9]*max(0,p[1]*input[2] + p[5])␣
↪+ p[10]*max(0,p[2]*input[2] + p[6]) + p[11]*max(0,p[3]*input[2] + p[7]) -␣
↪output[2],
    p[12] + p[8]*max(0,p[0]*input[3] + p[4]) + p[9]*max(0,p[1]*input[3] + p[5])␣
↪+ p[10]*max(0,p[2]*input[3] + p[6]) + p[11]*max(0,p[3]*input[3] + p[7]) -␣
↪output[3],
    p[12] + p[8]*max(0,p[0]*input[4] + p[4]) + p[9]*max(0,p[1]*input[4] + p[5])␣
↪+ p[10]*max(0,p[2]*input[4] + p[6]) + p[11]*max(0,p[3]*input[4] + p[7]) -␣
↪output[4],
    p[12] + p[8]*max(0,p[0]*input[5] + p[4]) + p[9]*max(0,p[1]*input[5] + p[5])␣
↪+ p[10]*max(0,p[2]*input[5] + p[6]) + p[11]*max(0,p[3]*input[5] + p[7]) -␣
↪output[5],
    p[12] + p[8]*max(0,p[0]*input[6] + p[4]) + p[9]*max(0,p[1]*input[6] + p[5])␣
↪+ p[10]*max(0,p[2]*input[6] + p[6]) + p[11]*max(0,p[3]*input[6] + p[7]) -␣
↪output[6],
    p[12] + p[8]*max(0,p[0]*input[7] + p[4]) + p[9]*max(0,p[1]*input[7] + p[5])␣
↪+ p[10]*max(0,p[2]*input[7] + p[6]) + p[11]*max(0,p[3]*input[7] + p[7]) -␣
↪output[7],
    p[12] + p[8]*max(0,p[0]*input[8] + p[4]) + p[9]*max(0,p[1]*input[8] + p[5])␣
↪+ p[10]*max(0,p[2]*input[8] + p[6]) + p[11]*max(0,p[3]*input[8] + p[7]) -␣
↪output[8],
    p[12] + p[8]*max(0,p[0]*input[9] + p[4]) + p[9]*max(0,p[1]*input[9] + p[5])␣
↪+ p[10]*max(0,p[2]*input[9] + p[6]) + p[11]*max(0,p[3]*input[9] + p[7]) -␣
↪output[9],
```

```
    p[12] + p[8]*max(0,p[0]*input[10] + p[4]) + p[9]*max(0,p[1]*input[10] +␣
 ↪p[5]) + p[10]*max(0,p[2]*input[10] + p[6]) + p[11]*max(0,p[3]*input[10] +␣
 ↪p[7]) - output[10],
    p[12] + p[8]*max(0,p[0]*input[10] + p[4]) + p[9]*max(0,p[1]*input[10] +␣
 ↪p[5]) + p[10]*max(0,p[2]*input[10] + p[6]) + p[11]*max(0,p[3]*input[10] +␣
 ↪p[7]) - output[10],
    p[12] + p[8]*max(0,p[0]*input[10] + p[4]) + p[9]*max(0,p[1]*input[10] +␣
 ↪p[5]) + p[10]*max(0,p[2]*input[10] + p[6]) + p[11]*max(0,p[3]*input[10] +␣
 ↪p[7]) - output[10]]
```

[6]:
```
initial_guess = np.random.normal(size =13)
solution = fsolve(f, initial_guess)
while not(np.all(np.isclose(f(solution),[0]*13)==True)):
  new_guess = np.random.normal(size =13)

  solution = fsolve(f, new_guess)
solution
```

[6]:
```
array([ -0.35614338,   0.05176055,   0.04679073,   4.26739128,
         1.06854914,   0.69438611,   0.33182292, -21.3355306 ,
        -5.61759265, -20.45828289,   1.24519253,   0.46882705,
        19.79543786])
```

[7]:
```
f(solution)
```

[7]:
```
[4.218847493575595e-15,
 7.105427357601002e-15,
 6.217248937900877e-15,
 3.552713678800501e-15,
 8.881784197001252e-16,
 4.440892098500626e-16,
 0.0,
 1.7763568394002505e-15,
 2.6645352591003757e-15,
 -8.881784197001252e-16,
 1.7763568394002505e-15,
 1.7763568394002505e-15,
 1.7763568394002505e-15]
```

[ ]: