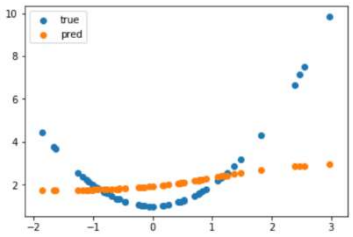
Objective: Assessing Sigmoid() as a substitute for ReLU()

Problem: Finding the best fit line for x^2 + 1 and higher degree polynomials

Attempt # 1: Started out with the standard 3 neurons and 3 layers (2 doesn't work at all for sigmoid)

```
a = 3
model = nn.Sequential(
    nn.Linear(1, a),
    nn.Sigmoid(),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

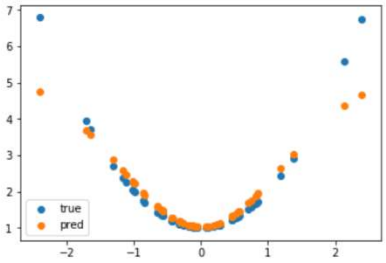
```
Epoch [500/10000], Loss: 2.0433
Epoch [1000/10000], Loss: 2.0322
Epoch [1500/10000], Loss: 2.0260
Epoch [2000/10000], Loss: 2.0199
Epoch [2500/10000], Loss: 2.0139
Epoch [3000/10000], Loss: 2.0079
Epoch [3500/10000], Loss: 2.0016
Epoch [4000/10000], Loss: 1.9950
Epoch [4500/10000], Loss: 1.9879
Epoch [5000/10000], Loss: 1.9803
Epoch [5500/10000], Loss: 1.9720
Epoch [6000/10000], Loss: 1.9630
Epoch [6500/10000], Loss: 1.9531
Epoch [7000/10000], Loss: 1.9422
Epoch [7500/10000], Loss: 1.9301
Epoch [8000/10000], Loss: 1.9169
Epoch [8500/10000], Loss: 1.9024
Epoch [9000/10000], Loss: 1.8864
Epoch [9500/10000], Loss: 1.8689
Epoch [10000/10000], Loss: 1.8498
```




Attempt # 2: increase the neurons to 20 but the split at the end is concerning

```
a = 20
model = nn.Sequential(
    nn.Linear(1, a),
    nn.Sigmoid(),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

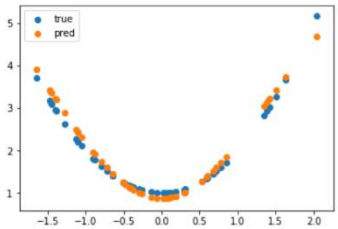
```
Epoch [500/10000], Loss: 1.8940
Epoch [1000/10000], Loss: 1.8764
Epoch [1500/10000], Loss: 1.8583
Epoch [2000/10000], Loss: 1.8391
Epoch [2500/10000], Loss: 1.8179
Epoch [3000/10000], Loss: 1.7939
Epoch [3500/10000], Loss: 1.7658
Epoch [4000/10000], Loss: 1.7320
Epoch [4500/10000], Loss: 1.6904
Epoch [5000/10000], Loss: 1.6382
Epoch [5500/10000], Loss: 1.5717
Epoch [6000/10000], Loss: 1.4862
Epoch [6500/10000], Loss: 1.3762
Epoch [7000/10000], Loss: 1.2370
Epoch [7500/10000], Loss: 1.0670
Epoch [8000/10000], Loss: 0.8725
Epoch [8500/10000], Loss: 0.6717
Epoch [9000/10000], Loss: 0.4920
Epoch [9500/10000], Loss: 0.3558
Epoch [10000/10000], Loss: 0.2683
```




Attempt # 3: try increasing the neurons but the result is far from acceptable

```
a = 100
model = nn.Sequential(
    nn.Linear(1, a),
    nn.Sigmoid(),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

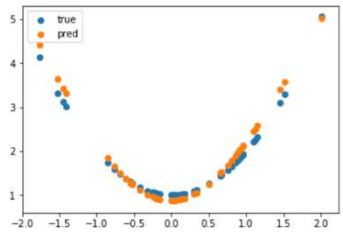
```
Epoch [500/10000], Loss: 1.7006
Epoch [1000/10000], Loss: 1.6718
Epoch [1500/10000], Loss: 1.6427
Epoch [2000/10000], Loss: 1.6103
Epoch [2500/10000], Loss: 1.5724
Epoch [3000/10000], Loss: 1.5265
Epoch [3500/10000], Loss: 1.4693
Epoch [4000/10000], Loss: 1.3971
Epoch [4500/10000], Loss: 1.3055
Epoch [5000/10000], Loss: 1.1902
Epoch [5500/10000], Loss: 1.0484
Epoch [6000/10000], Loss: 0.8812
Epoch [6500/10000], Loss: 0.6978
Epoch [7000/10000], Loss: 0.5170
Epoch [7500/10000], Loss: 0.3621
Epoch [8000/10000], Loss: 0.2500
Epoch [8500/10000], Loss: 0.1815
Epoch [9000/10000], Loss: 0.1452
Epoch [9500/10000], Loss: 0.1277
Epoch [10000/10000], Loss: 0.1191
```




Attempt # 4: Tried increasing the neurons before I started messing around with the layers

```
model = nn.Sequential(
    nn.Linear(in_dim, 1000),
    nn.Sigmoid(),
    nn.Linear(1000,1000),
    nn.Linear(1000, out_dim)
)
```

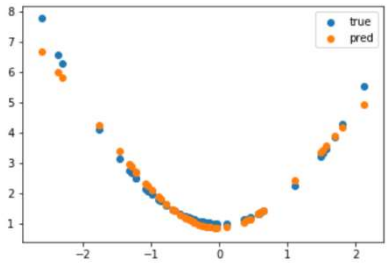
```
Epoch [500/10000], Loss: 1.4862
Epoch [1000/10000], Loss: 1.0326
Epoch [1500/10000], Loss: 0.5187
Epoch [2000/10000], Loss: 0.1951
Epoch [2500/10000], Loss: 0.0900
Epoch [3000/10000], Loss: 0.0688
Epoch [3500/10000], Loss: 0.0652
Epoch [4000/10000], Loss: 0.0644
Epoch [4500/10000], Loss: 0.0737
Epoch [5000/10000], Loss: 0.0636
Epoch [5500/10000], Loss: 0.0631
Epoch [6000/10000], Loss: 0.0627
Epoch [6500/10000], Loss: 0.0624
Epoch [7000/10000], Loss: 0.0679
Epoch [7500/10000], Loss: 0.0630
Epoch [8000/10000], Loss: 0.0614
Epoch [8500/10000], Loss: 0.0613
Epoch [9000/10000], Loss: 0.0732
Epoch [9500/10000], Loss: 0.0627
Epoch [10000/10000], Loss: 0.0610
```



Attempt # 5: 4 layers with 100 neurons (low # of neurons just doesn't work)

```
a = 100
model = nn.Sequential(
    nn.Linear(1, a),
    nn.Sigmoid(),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

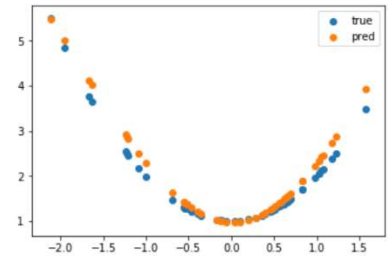
```
Epoch [500/10000], Loss: 1.4432
Epoch [1000/10000], Loss: 1.4216
Epoch [1500/10000], Loss: 1.3973
Epoch [2000/10000], Loss: 1.3678
Epoch [2500/10000], Loss: 1.3288
Epoch [3000/10000], Loss: 1.2741
Epoch [3500/10000], Loss: 1.1925
Epoch [4000/10000], Loss: 1.0659
Epoch [4500/10000], Loss: 0.8687
Epoch [5000/10000], Loss: 0.5866
Epoch [5500/10000], Loss: 0.2822
Epoch [6000/10000], Loss: 0.0959
Epoch [6500/10000], Loss: 0.0411
Epoch [7000/10000], Loss: 0.0317
Epoch [7500/10000], Loss: 0.0300
Epoch [8000/10000], Loss: 0.0292
Epoch [8500/10000], Loss: 0.0285
Epoch [9000/10000], Loss: 0.0279
Epoch [9500/10000], Loss: 0.0273
Epoch [10000/10000], Loss: 0.0268
```




Attempt # 6: increase in layers yields worse result (6 layers for this case)

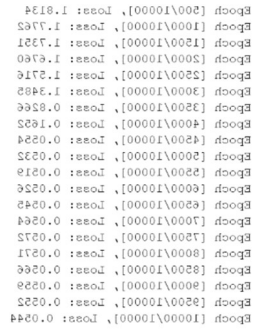
```
a = 100
model = nn.Sequential(
    nn.Linear(1, a),
    nn.Sigmoid(),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

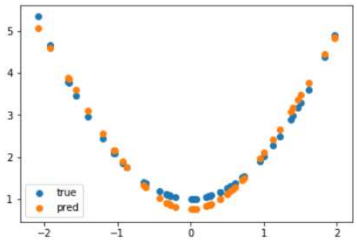
```
Epoch [500/10000], Loss: 2.1436
Epoch [1000/10000], Loss: 2.1175
Epoch [1500/10000], Loss: 2.0838
Epoch [2000/10000], Loss: 2.0315
Epoch [2500/10000], Loss: 1.9260
Epoch [3000/10000], Loss: 1.6136
Epoch [3500/10000], Loss: 0.4867
Epoch [4000/10000], Loss: 0.1292
Epoch [4500/10000], Loss: 0.0867
Epoch [5000/10000], Loss: 0.2299
Epoch [5500/10000], Loss: 0.0935
Epoch [6000/10000], Loss: 0.1021
Epoch [6500/10000], Loss: 0.0965
Epoch [7000/10000], Loss: 0.0931
Epoch [7500/10000], Loss: 0.0908
Epoch [8000/10000], Loss: 0.0887
Epoch [8500/10000], Loss: 0.0870
Epoch [9000/10000], Loss: 0.0854
Epoch [9500/10000], Loss: 0.0839
Epoch [10000/10000], Loss: 0.0825
```




Attempt # 7: just to make sure 4 layers is the best, I checked 5 layers

```
a = 100
model = nn.Sequential(
    nn.Linear(1, a),
    nn.Sigmoid(),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

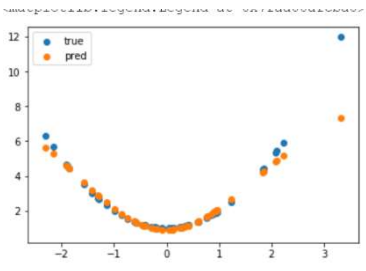





Attempt # 8: increase in neurons doesn't improve the result but isn't worth the time that it takes and graphs also diverge at the end for Sigmoid()

```
a = 300
model = nn.Sequential(
    nn.Linear(1, a),
    nn.Sigmoid(),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

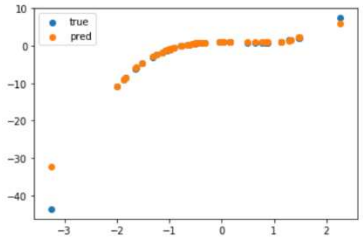
```
Epoch [500/10000], Loss: 1.1389
Epoch [1000/10000], Loss: 1.0912
Epoch [1500/10000], Loss: 1.0282
Epoch [2000/10000], Loss: 0.9326
Epoch [2500/10000], Loss: 0.7790
Epoch [3000/10000], Loss: 0.5446
Epoch [3500/10000], Loss: 0.2700
Epoch [4000/10000], Loss: 0.0875
Epoch [4500/10000], Loss: 0.0311
Epoch [5000/10000], Loss: 0.0218
Epoch [5500/10000], Loss: 0.0207
Epoch [6000/10000], Loss: 0.0205
Epoch [6500/10000], Loss: 0.0204
Epoch [7000/10000], Loss: 0.0203
Epoch [7500/10000], Loss: 0.0202
Epoch [8000/10000], Loss: 0.0202
Epoch [8500/10000], Loss: 0.0201
Epoch [9000/10000], Loss: 0.0200
Epoch [9500/10000], Loss: 0.0199
Epoch [10000/10000], Loss: 0.0199
```



Attempt # 9: Went back to ReLU() and see if the best set up for polynomial of degree 2 works for higher degree as well. Function f(x) = x^3 – x ^2 -1

```
a = 20
model = nn.Sequential(
    nn.Linear(1, a),
    nn.ReLU(),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

```
Epoch [500/10000], Loss: 3.6529
Epoch [1000/10000], Loss: 0.8467
Epoch [1500/10000], Loss: 0.3457
Epoch [2000/10000], Loss: 0.1781
Epoch [2500/10000], Loss: 0.0957
Epoch [3000/10000], Loss: 0.0548
Epoch [3500/10000], Loss: 0.0344
Epoch [4000/10000], Loss: 0.0252
Epoch [4500/10000], Loss: 0.0217
Epoch [5000/10000], Loss: 0.0204
Epoch [5500/10000], Loss: 0.0199
Epoch [6000/10000], Loss: 0.0197
Epoch [6500/10000], Loss: 0.0193
Epoch [7000/10000], Loss: 0.0185
Epoch [7500/10000], Loss: 0.0188
Epoch [8000/10000], Loss: 0.0213
Epoch [8500/10000], Loss: 0.0258
Epoch [9000/10000], Loss: 0.0279
Epoch [9500/10000], Loss: 0.0246
Epoch [10000/10000], Loss: 0.0214
```




Attempt # 10: Function 3x^4+ x^3 – x ^2 -1 gave the same result

```
a = 20
model = nn.Sequential(
    nn.Linear(1, a),
    nn.ReLU(),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, a),
    nn.Linear(a, 1)
)
```

```
Epoch [500/10000], Loss: 50.0880
Epoch [1000/10000], Loss: 20.4932
Epoch [1500/10000], Loss: 9.9422
Epoch [2000/10000], Loss: 5.6731
Epoch [2500/10000], Loss: 3.7136
Epoch [3000/10000], Loss: 2.7464
Epoch [3500/10000], Loss: 2.2200
Epoch [4000/10000], Loss: 1.9351
Epoch [4500/10000], Loss: 1.7789
Epoch [5000/10000], Loss: 1.6893
Epoch [5500/10000], Loss: 1.6355
Epoch [6000/10000], Loss: 1.6049
Epoch [6500/10000], Loss: 1.5874
Epoch [7000/10000], Loss: 1.5780
Epoch [7500/10000], Loss: 1.5725
Epoch [8000/10000], Loss: 1.5695
Epoch [8500/10000], Loss: 1.5678
Epoch [9000/10000], Loss: 1.5670
Epoch [9500/10000], Loss: 1.5666
Epoch [10000/10000], Loss: 1.5664
```