IT시스템설계

과제2: Numpy와 Tensorflow 1.x로 Neural Network 구현

전자전기공학부
B415156 이승호

1. Neural Network (using numpy)

Numpy로 간단한 Neural Network를 Python Class 형식으로 구현했다. Input, Output Node의 수와 Hidden Layer, Hidden Node의 수와 Activation Function을 변경할 수 있게 만들었다. 아쉬운 것은 Hidden Node의 개수는 모든 Layer에 대해서 하나로 밖에 고정할 수 없다는 점이다. Learning Rate, # of Hidden Layer, Hidden Node, Iteratoin을 바꾸어가며 성능의 차이를 확인했다.

Full Adder의 진리값에서 False -> -1로 매칭하고 나니 Sigmoid Function으로는 학습이 잘 되지 않아 Tanh Function을 사용했다.

1) Learning Rate가 수렴에 미치는 영향

- # of Hidden Layer: 1

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	0.3217	1.0000
0.1	0.0345	1.0000
0.3	0.0190	1.0000
0.5	0.0148	1.0000

- # of Hidden Layer: 2

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	0.1501	1.0000
0.1	0.0194	1.0000
0.3	0.0125	1.0000
0.5	0.0093	1.0000

- # of Hidden Layer: 3

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	0.6770	0.9838
0.1	0.0230	1.0000
0.3	0.0109	1.0000
0.5	0.0102	1.0000

- # of Hidden Layer: 4

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	2.0006	0.4925
0.1	0.0350	1.0000
0.3	0.7650	0.7500
0.5	0.7684	0.7475

대체로 Layer의 Depth가 낮은 경우 대체로 모든 Learning Rate에서 학습이 잘 되었다. Layer가 3단일 때 까지는 Learning Rate를 증가시킬수록 학습이 빠르게 이루어졌다. 그러나 Hidden Layer를 4단까지 쌓은 경우 특정 Learning Rate에서만 학습이 이루어지는 민감한 모습을 볼 수 있었다.

2) 입력신호의 Variance가 Weight에 미치는 영향

입력신호의 Noise Variance가 Weight에 무슨 영향을 미치는지 잘 모르겠어서 아래와 같은 방식으로 변화를 테스트해보았다.

```
1. temp = abs(Initial_Weight - Trained_Weight)
2. result = sum(temp)
```

# of Hidden Layer: 1

# of Hidden Node: 3

Epoch: 1000

Variance	0	0.1	0.3	0.5
W_Hidden	8.97	11.04	11.04	11.04
W_Output	9.62	9.58	9.58	9.58
B_Hidden	0.00	0.00	0.00	0.00
B_Output	0.06	0.10	0.10	0.10

Epoch: 10000

Variance	0	0.1	0.3	0.5
W_Hidden	12.30	11.04	11.04	11.04
W_Output	14.28	9.58	9.58	9.58
B_Hidden	0.00	0.00	0.00	0.00
B_Output	0.20	0.10	0.10	0.10

Test Data의 Noise Variance가 없을 때와 있을 때는 값이 달랐지만 Noise가 있을 때부터는 값이 딱히 변하지 않았다. Epoch의 수를 늘려도 Weight와 Bias가 변하긴 하지만 Variance가 생기고 나서부터는 그 값이 늘어도 변하지 않았다.

3) Epoch 횟수에 따른 Error, Accuracy의 변화

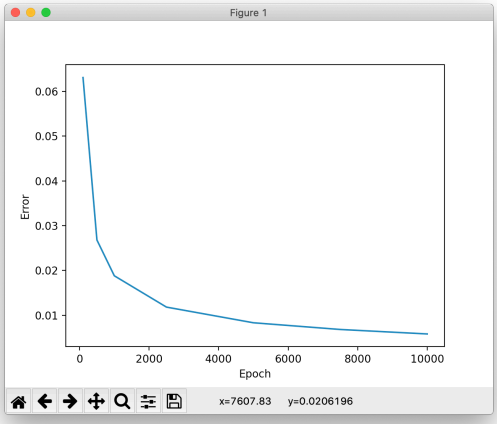
of Hidden Layer: 1

# of Hidden Node: 3

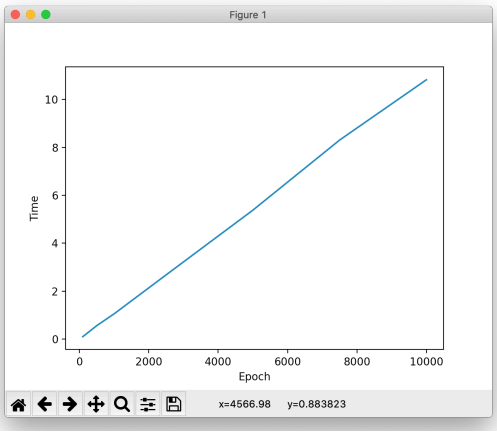
Learning Rate: 0.3

Test Noise Variance: 0.1

Epoch	Error	Accuracy	Time [s]
100	0.0631	1.0000	0.106
500	0.0268	1.0000	0.564
1000	0.0188	1.0000	1.056
2500	0.0118	1.0000	2.682
5000	0.0083	1.0000	5.380
7500	0.0068	1.0000	8.303
10000	0.0058	1.0000	10.817



[Plot: x-Epoch, y-Error Plot]



[Plot: x-Epoch, y-Processing Time]

Epoch의 증가에 따라 Error는 Exponential하게 감소되고 Processing Time은 Linear하게 증가됨을 확인할 수 있었다.

4) Hidden Layer, Node의 수가 성능에 미치는 영향

Learning Rate: 0.3

Test Noise Variance: 0.1

Epoch: 2000

*Error

Layer / Node	1	2	3	4
1	0.7370	0.7524	0.7540	0.7624
2	0.0201	0.7304	0.3821	0.3715
3	0.0143	0.0084	0.0086	0.7150
4	0.0112	0.0082	0.0091	0.7699

*Accuracy

Layer / Node	1	2	3	4
1	0.7500	0.7500	0.7500	0.7500
2	1.0000	0.7325	0.8750	0.8750
3	1.0000	1.0000	1.0000	0.8588
4	1.0000	1.0000	1.0000	0.7488

Node가 1개일때는 공통적으로 학습이 되지 않았고 Layer가 너무 늘어난 경우에도 학습이 잘 되지 않았다. 적절한 Hyper Parameter의 수를 잘 맞춰야할 것 같다.

2. Neural Network (using tensorflow)

Tensorflow로도 같은 구조로 Neural Network를 만들어서 다양한 테스트를 해보았다. Training에 사용한 Optimizer는 GradientDescentOptimizer를 이용했다. Activation Function은 모든 Layer에서 Tanh Function을 이용했다.

1) Learning Rate가 수렴에 미치는 영향

- # of Hidden Layer: 1

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	0.5991	0.8750
0.1	0.0230	1.0000
0.3	0.0058	1.0000
0.5	0.0033	1.0000

- # of Hidden Layer: 2

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	0.9180	0.6875
0.1	0.0557	1.0000
0.3	0.0066	1.0000
0.5	0.0050	1.0000

- # of Hidden Layer: 3

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	1.4407	0.5000
0.1	0.0111	1.0000
0.3	0.0004	1.0000
0.5	0.1680	0.9375

- # of Hidden Layer: 4

# of Hidden Node: 3

Epoch: 1000

Test Noise Variance = 0.1

Learning Rate	Error	Accuracy
0.01	0.1493	0.8750
0.1	0.0132	1.0000
0.3	0.0015	1.0000
0.5	0.0002	1.0000

Learning Rate가 너무 작은 경우를 제외한 대부분의 경우에서 학습이 잘 이루어졌다. Numpy로 만든 Neural Network와 달리 Layer가 4단 쌓였어도 학습이 잘 되었다.

2) 입력신호의 Variance가 Weight에 미치는 영향

Numpy와 마찬가지의 측정 방식으로 Weight의 변화량의 합을 측정했다.

# of Hidden Layer: 1

# of Hidden Node: 3

Epoch: 1000

Variance	0	0.1	0.3	0.5
W_Hidden	8.00	10.93	6.18	9.38
W_Output	7.41	8.85	8.19	6.26
B_Hidden	2.07	1.17	3.02	1.82
B_Output	2.19	1.72	0.85	1.63

Epoch: 10000

Variance	0	0.1	0.3	0.5
W_Hidden	8.30	5.34	8.79	10.29
W_Output	12.11	11.85	10.40	13.69
B_Hidden	1.21	2.03	0.56	4.56
B_Output	2.61	1.29	1.66	1.30

Numpy의 경우와 달리 Test Input의 Noise Variance를 바꾸니 Weight와 Bias가 계속 변화했다. Weight과 Bias의 변화가 무엇을 의미하는지는 잘 모르겠다.

3) Epoch 횟수에 따른 Error, Accuracy의 변화

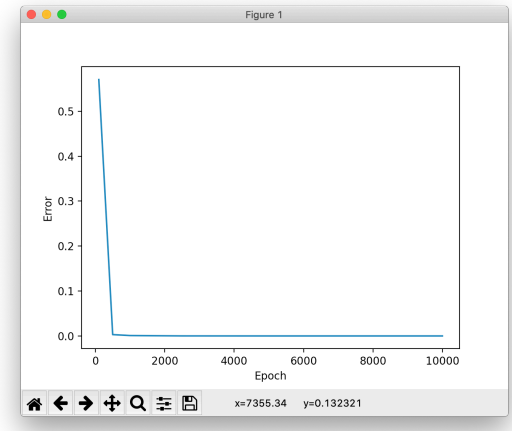
of Hidden Layer: 1

# of Hidden Node: 3

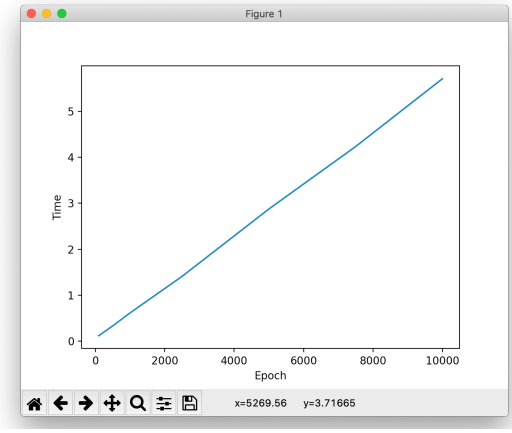
Learning Rate: 0.3

Test Noise Variance: 0.1

Epoch	Error	Accuracy	Time [s]
100	0.5704	0.6875	0.119
500	0.0031	1.0000	0.331
1000	0.0010	1.0000	0.614
2500	0.0002	1.0000	1.409
5000	0.0001	1.0000	2.873
7500	0.0001	1.0000	4.231
10000	0.0001	1.0000	5.705



[Plot: x-Epoch, y-Error Plot]



[Plot: x-Epoch, y-Processing Time]

Numpy에 비해 Epoch가 증가하면서 Error는 더욱 많이 감소되었고 Processing Time은 마찬가지로 Linear하게 증가되었다. 차이점은 Numpy로 구현한 Neural Network에 비해 Processing Time이 2배 가까이 빨랐다.

4) Hidden Layer, Node의 수가 성능에 미치는 영향

Learning Rate: 0.3

Test Noise Variance: 0.1

Epoch: 2000

*Error

Layer / Node	1	2	3	4
1	0.3763	0.8754	0.3756	0.3752
2	0.0035	0.2505	0.1266	0.3745
3	0.0004	0.0005	0.0001	0.0004
4	0.0004	0.0006	0.0004	0.0003

*Accuracy

Layer / Node	1	2	3	4
1	0.8750	0.6250	0.8750	0.8750
2	1.0000	0.8125	0.8750	0.8750
3	1.0000	1.0000	1.0000	0.9375
4	1.0000	1.0000	1.0000	1.0000

마찬가지로 Node가 1개일때는 공통적으로 학습이 되지 않았다. 그러나 Layer가 4단이어도 학습이 잘 되었다.