고급 소프트웨어 실습1 보고서

5주차

월요일 분반

20211530 김희진

0. 실험목표

우리는 digit image를 식별하는 MLP classification model을 디자인하고 이에 대해 분석할 것이다. activation function과 code에 대한 기본 이해를 바탕으로 activation function, learning rate, depth of layers and hidden layer dimension 4가지에 대해 실험한 결과와 이론을 결부해 분석할 것이다.

1. 배경지식

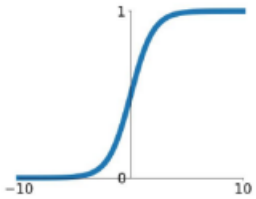
딥러닝 네트워크에서 노드에 입력된 값들은 비선형 함수에 통과되어 다음 레이어로 전달된다. 이때 사용되는 비선형 함수를 Activation function이라 한다. Activation function은 input data를 next layer로 어떻게 보낼 것인지를 전달하기에 매우 중요하다. 그러므로 해당 보고서에서 activation function을 위주로 실험결과를 분석할 것이고 분석에 사용될 activation function 4가지를 살펴볼 것이다.

1. Sigmoid function

Sigmoid 함수는 logistic 함수라고 불리기도 하며, x의 값에 따라 0~1의 값을 출력하는 S자형 함수이다. 확률을 출력하기 때문에 결과를 확률로 해석할 때 유용하다. 정의와 그래프는 아래와 같다.

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Back propagation으로 학습하는 방식은 chain rule을 이용하는데, 그래프를 보면 Sigmoid function은 x의 절댓값이 커질수록 미분 시 굉장히 작은 값을 도출한다. 이로 인해 sigmoid함수는 최종 layer에 미치는 영향이 적어지는 문제점이 발생할 수 있다. 다시 말해, hidden layer의 깊이가 깊으면 오차율을 계산하기 어렵기에 x의 절댓값이 커질수록 gradient backpropagation 시 미분 값이 소실될 가능성이 있다. 이렇게 layer가 deep할수록 기울기가 사라져 버리는 것을 gradient vanishing이라 한다.

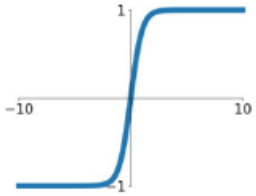
또한 함수의 중심점이 0이 아니다. 즉, x=0일 때 y의 값이 0이 아닌데 이는 같은 방향으로 업데이트되는 학습을 지그재그로 만들어 버려 속도를 느리게 한다.

2. Tanh function

Sigmoid함수를 변형해서 얻은 함수로 정의와 그래프는 아래와 같다.

tanh

$$\tanh(x)$$



Tanh function은 중심점을 0으로 옮겨 sigmoid function이 가지고 있던 최적화 과정에서 느려진다는 단점을 해결했다. 하지만 x의 절댓값이 커질수록 gradient backpropagation 시 미분 값이 소실될 가능성은 여전히 줄이지 못했다. 하지만 tanh function은 sigmoid function보다 기울기가 크기 때문에 gradient vanishing문제가 여전히 있긴 하지만 sigmoid보다 덜하다.

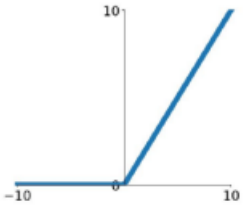
3. Relu function

Relu는 가장 많이 사용되는 activation function 중 하나로 sigmoid와 tanh function이 갖는 문제(gradient vanishing)를 해결하기 위한 함수이다.

정의와 그래프는 아래와 같다.

ReLU

$$\max(0, x)$$



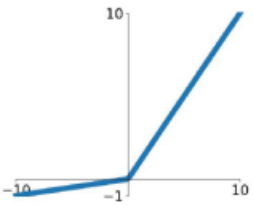
X가 0보다 크면 기울기가 1인직선, 0보다 작으면 함수값이 0이된다. 이런 특징으로 sigmoid, tanh function보다 학습이 빠르고 연산비용이 적고 구현이 간단하다는 특징이 있지만 x가 0이하의 수일 때 함수값이 0이므로 이때, 뉴런이 죽을 수 있다는 단점이 있다.

4. Leaky Relu with c=0.01

Relu가 갖는 뉴런이 죽는 상황을 해결하기 위해 나온 함수이다. 정의와 그래프는 아래와 같다.

Leaky ReLU

$$\max(0.1x, x)$$

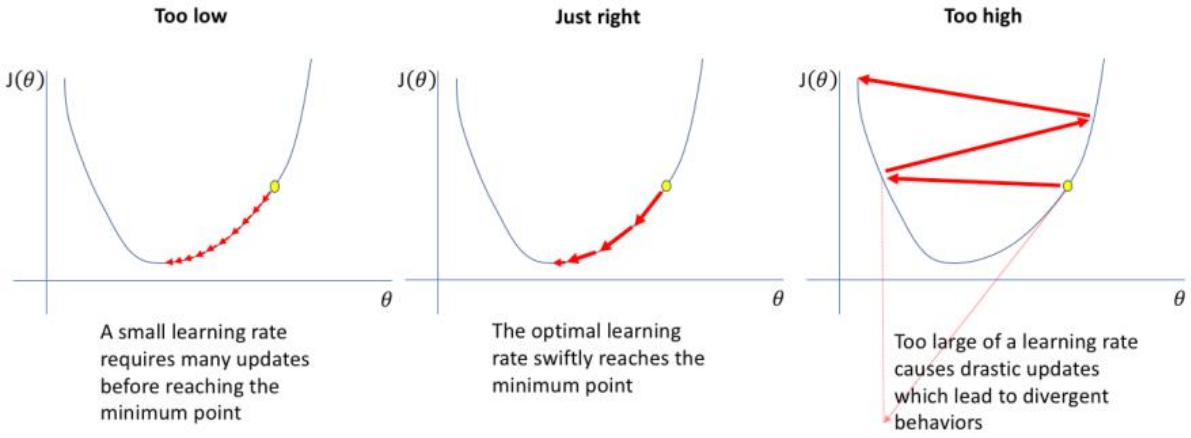


이때 x가 0이하의 수일 때 그래프의 기울기는 0.1뿐만아니라 매우 작은 값이라면 무엇이든 사용 가능하다. 해당 보고서에선 0.01에 대해서 분석할 것이다. Leaky relu function은 x가 음수인 영역의 값에 대해 미분값이 0이되지 않는다는 점을 제외하면 relu function의 특징을 동일하게 갖는다.

5. Learning rate

2번째 analysis에서 최고의 성능을 갖는 learning rate를 activation function별로 찾아줄 것이다. 그 전에, learning rate가 무엇인지에 대해 알아보자.

Learning rate는 gradient descent algorithm에서 한번 경사를 내려갈 때 얼만큼 움직일지에 대한 수치이다. 이는 절대적이지 않으며 환경에 따라 적절히 변경해주어야 한다.



너무 큰 learning rate를 설정하게 된다면 위 그림의 가장 오른쪽 그림처럼 움직이게 되어 minimum point에 도달하기에 너무 큰 udpate를 유발한다. 너무 작은 learning rate를 설정하게 된다면 위 그림의 가장 왼쪽 그림처럼 움직이게 되어 minimum point에 도달하기까지 너무 많은 이동이 필요하다.

그렇기 때문에 우리는 적당한 learning rate를 찾는 것이 중요하다.

2. Analysis

1) Analysis-Activation function

Activation function에 따른 average loss, accuracy, total training and evaluation time을 분석해보자.

a. Relu

-epoch 1

```
Test set: Average loss: 0.8641, Accuracy: 8038/10000 (80%)
```

-epoch 2

```
Test set: Average loss: 0.5120, Accuracy: 8665/10000 (87%)
```

-epoch 3

```
Test set: Average loss: 0.4321, Accuracy: 8822/10000 (88%)
```

-epoch 4

```
Test set: Average loss: 0.3983, Accuracy: 8897/10000 (89%)
```

-epoch 5

```
Test set: Average loss: 0.3805, Accuracy: 8941/10000 (89%)
```

-total training and evaluation time

```
Total training and evaluation time: 85.3004 초
```

b. Tanh

-epoch 1

```
Test set: Average loss: 1.1195, Accuracy: 7749/10000 (77%)
```

-epoch 2

```
Test set: Average loss: 0.7508, Accuracy: 8583/10000 (86%)
```

-epoch 3

```
Test set: Average loss: 0.6148, Accuracy: 8757/10000 (88%)
```

-epoch 4

```
Test set: Average loss: 0.5509, Accuracy: 8839/10000 (88%)
```

-epoch 5

```
Test set: Average loss: 0.5161, Accuracy: 8882/10000 (89%)
```

-total training and evaluation time

```
Total training and evaluation time: 89.7824 초
```

c. Sigmoid

-epoch 1

```
Test set: Average loss: 2.2919, Accuracy: 1718/10000 (17%)
```

-epoch 2

```
Test set: Average loss: 2.2835, Accuracy: 1663/10000 (17%)
```

-epoch 3

```
Test set: Average loss: 2.2769, Accuracy: 1697/10000 (17%)
```

-epoch 4

```
Test set: Average loss: 2.2717, Accuracy: 1750/10000 (18%)
```

-epoch 5

```
Test set: Average loss: 2.2678, Accuracy: 1830/10000 (18%)
```

-total training and evaluation time

```
Total training and evaluation time: 98.6995 초
```

d. LeakyRelu with c=0.01

-epoch 1

```
Test set: Average loss: 0.8620, Accuracy: 8039/10000 (80%)
```

-epoch 2

```
Test set: Average loss: 0.5113, Accuracy: 8664/10000 (87%)
```

-epoch 3

```
Test set: Average loss: 0.4318, Accuracy: 8823/10000 (88%)
```

-epoch 4

```
Test set: Average loss: 0.3982, Accuracy: 8902/10000 (89%)
```

-epoch 5

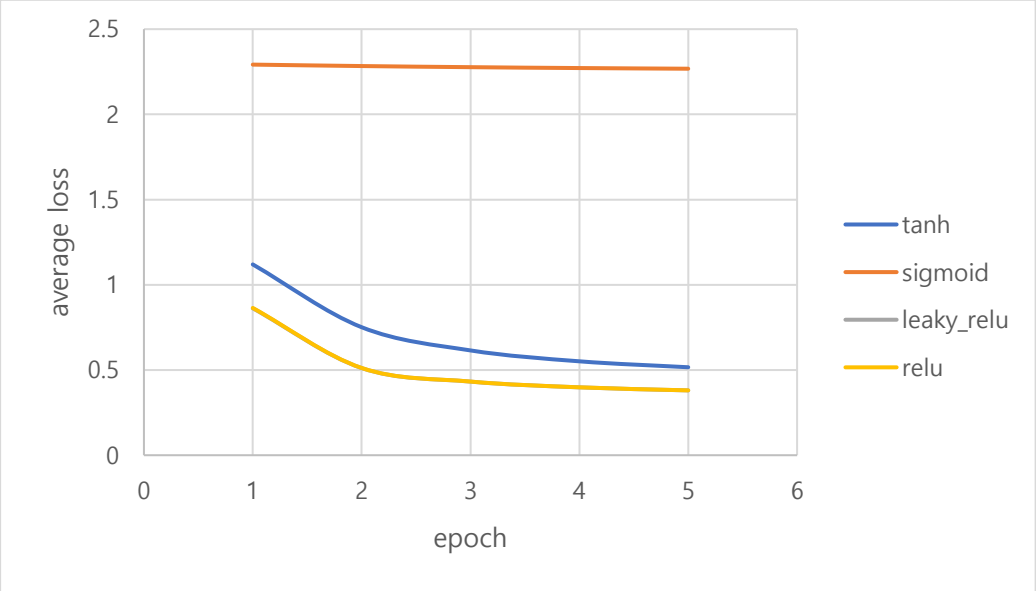
```
Test set: Average loss: 0.3804, Accuracy: 8941/10000 (89%)
```

-total training and evaluation time

```
Total training and evaluation time: 89.1866 초
```

e. Comparation for average loss

위의 첨부한 결과들을 각 함수별로 average loss에 대해 그래프로 나타내면 다음과 같다.

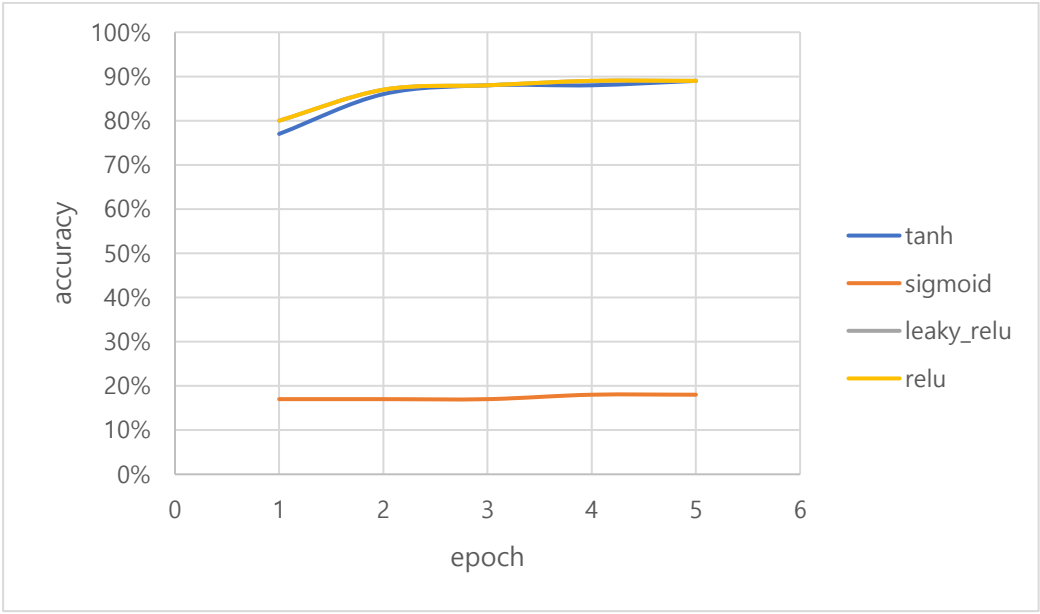


모든 epoch에 대해 sigmoid>tanh>leaky_relu=relu순으로 loss가 크다. 그래프에서 알 수 있듯 sigmoid와 tanh는 2이상 4이하 정도로 loss가 차이가 나지만 tanh와 relu(leaky_relu)는 1이하 정도로 loss가 차이난다. 또한, relu와 leaky_relu는 거의 비슷한 average loss를 가진다.

4가지 activation function은모두 epoch 1에서 가장 높은 average loss를 기록하며 epoch 2, 3, 4, 5의 순서대로 average loss는 낮아진다. 즉, epoch 5, 4, 3, 2, 1의 순서로 loss가 덜하게 학습할 수 있는 데이터이다. Sigmoid를 제외한 3함수에서는 epoch 1과 epoch 2의 차이가 뚜렷하지만 sigmoid에서는 다른 함수에 비해 두드러지지 않는다.

1.배경지식에서 언급했듯 Sigmoid와 tanh function 둘 다 x의 절댓값이 커질수록 미분값이 작아지기 때문에 leaky_relu와 relu function보다 average loss가 큰 것을 실험결과로 확인할 수 있다. 또한, tanh function은 sigmoid function과 비슷한 모양을 띄는 함수지만 미분값이 더 크기 때문에 sigmoid function보다 average loss보다 적음을 확인할 수 있다.

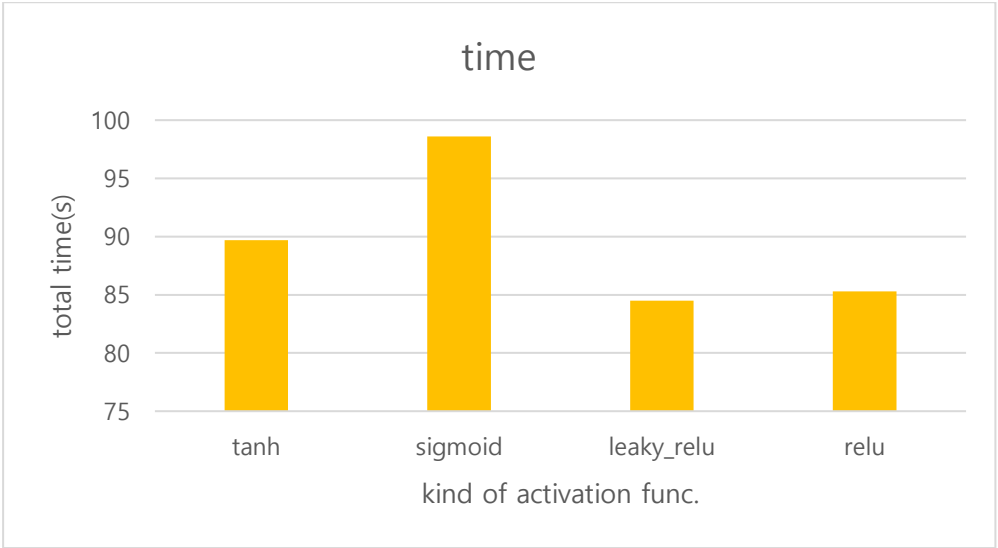
f. Comparation for accuracy



모든 epoch에 대해 sigmoid는 10%에서 20%사이의 accuracy를 가지지만 다른 3가지 함수는 75%에서 90%사이의 accuracy를 가진다.

또한 sigmoid를 제외한 3개의 함수에서는 epoch1보다 epoch2가 높은 accuracy를 가지며 이가 두드러지게 나타난다.

g. Comparation total training and evaluation time



Total training and evaluation time을 비교했을 때 tanh>sigmoid>relu>leaky_relu순으로 시간이 많이 들었다. Accuracy나 average loss면에서 sigmoid를 제외한 3함수가 비슷했지만 시간적인 측면에서는 leaky_relu와 relu가 tanh보다 시간이 훨씬 적게 들었다.

1.배경지식에서 언급했듯 tanh함수는 중심점을 0으로 옮겨 sigmoid함수의 속도를 개선한 함수이다. 다시 말해, sigmoid함수는 중심점이 0이 아니기에 중심점이 0인 함수보다 느리다. 이를 위 그래프에서 확인할 수 있다. 또한 relu와 leaky_relu함수는 input data x가 0이상일 때 기울기가 1인 직선이므로 tanh, sigmoid 함수보다 속도가 빠른데 이도 실험결과에서 확인할 수 있었다.

2) Analysis-Learning rate for each activation function

Learning rate를 변경하여 average loss, accuracy, training and evaluation total time 측면에서 최고의 성능을 내는 learning rate를 찾아볼 것이다. 이때 average loss와 accuracy는 epoch1~5의 각각의 값들을 평균내 비교해 주었다.

a. Relu

- Comparation for average loss

learning rate	average loss
0.001	1.399
0.01	0.286
0.03	0.517
0.1	0.124
0.12	0.126
0.13	0.120
0.14	0.121
0.15	0.121
0.16	0.119
0.17	0.119
0.2	0.133

위 실험 결과로 average loss는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.1~0.2사이에서 average loss가 크게 주는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았을 때 0.16~0.17구간에서 가장 작은 average loss를 가짐을 확인했다. 하지만 average loss는 같은 값을 실험하더라도 달라지기에 이를 섣불리 결론 내리기 어렵다고 생각한다. 즉, 결론은 0.1과 0.2사이에서 가장 작은 average loss를 가질 것이라는 것이다.

- Comparation for accuracy

learning rate	accuracy

0.001	2.7
0.01	91.6
0.03	86.6
0.1	96.2
0.12	96
0.13	96
0.14	96
0.15	96.2
0.16	96.2
0.17	96
0.2	96.2

위 실험 결과로 accuracy는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.1이상에서 accuracy가 크게 느는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았지만, accuracy는 정수로 각 epoch에 대해 %로만 결과가 나오기 때문에 정확한 판단을 하기 어려운 지표라고 생각했다. 위 average loss의 결과와 함께 생각해보았을 때 relu function은 0.1과 0.2사이의 learning rate에서 가장 좋은 성능을 낸다고 평가할 수 있다.

- Comparation for total training and evaluation time

learning rate	time
0.001	86.2759
0.01	224.8384
0.03	85.3004
0.1	217.86
0.12	96.3
0.13	192.878
0.14	87.9623
0.15	89.9051
0.16	228.9454
0.17	190.219
0.2	232.227

위 실험 결과로 total time은 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. Accuracy와 average loss는 거의 반비례 관계를 가졌지만 time은 이 둘과 반비례 혹은 비례관계를 찾기 어려웠다.

앞서 최적의 성능을 내는 learning rate로 예상했던 0.1과 0.2사이구간에서 거의 200초대인 다른 rate와 다르게 0.14와 0.1는 80초대의 total time을 가진다.

앞선 세 지표로 인한 결과는 0.14~0.15사이의 learning rate일 때 relu함수는 최적의 성능을 낼 수 있다는 것이다.

b. Tanh

- Comparation for average loss

learning rate	average loss
0.001	1.41618
0.01	0.3353
0.03	0.71042
0.1	0.1287
0.13	0.1229
0.15	0.1195
0.17	0.1193
0.18	0.1188
0.19	0.1225
0.2	0.1256
0.3	0.1328

위 실험 결과로 average loss는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.1~0.2사이에서 average loss가 크게 주는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았을 때 0.17~0.18구간에서 가장 작은 average loss를 가짐을 확인했다. 하지만 average loss는 같은 값을 실험하더라도 달라지기에 이를 섣불리 결론 내리기 어렵다고 생각한다. 즉, 결론은 0.1과 0.2사이에서 가장 작은 average loss를 가질 것이라는 것이다.

- Comparation for accuracy

learning rate	accuracy
0.001	71.4
0.01	91.2
0.03	85.6

0.1	96
0.13	96.4
0.15	96.4
0.17	96.4
0.18	96.4
0.19	96.4
0.2	96.4
0.3	95.8

위 실험 결과로 accuracy는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.1이상에서 accuracy가 크게 느는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았지만, accuracy는 정수로 각 epoch에 대해 %로만 결과가 나오기 때문에 정확한 판단을 하기 어려운 지표라고 생각했다. 위 average loss의 결과와 함께 생각해보았을 때 relu function은 0.1과 0.2사이의 learning rate에서 가장 좋은 성능을 낸다고 평가할 수 있다.

- Comparation for total training and evaluation time

learning rate	time
0.001	94.3781
0.01	88.546
0.03	98.6995
0.1	153.2248
0.13	99.331
0.15	94.3868
0.17	96.0095
0.18	88.4803
0.19	87.4772
0.2	95.232
0.3	93.2837

위 실험 결과로 total time은 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. Accuracy와 average loss는 거의 반비례 관계를 가졌지만 time은 이 둘과 반비례 혹은 비례관계를 찾기 어려웠다.

앞서 최적의 성능을 내는 learning rate로 예상했던 0.1과 0.2사이구간에서 90초대인 다른 rate와 다르게 0.18와 0.19는 80초대의 total time을 가진다.

앞선 세 지표로 인한 결과는 0.18~0.19사이의 learning rate일 때 tanh함수는 최적의 성능을 낼 수 있다는 것이다.

c. Sigmoid

- Comparation for average loss

learning rate	average loss
0.001	2.2962
0.01	2.0671
0.03	0.8
0.1	0.3471
0.2	0.2261
0.3	0.1845
0.4	0.1625
0.5	0.1495
0.6	0.1418
0.61	0.1412
0.62	0.1406
0.63	0.1411
0.65	0.1478
0.7	0.1427

위 실험 결과로 average loss는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.6~0.7사이에서 average loss가 크게 주는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았을 때 0.62~0.63구간에서 가장 작은 average loss를 가짐을 확인했다. 하지만 average loss는 같은 값을 실험하더라도 달라지기에 이를 섣불리 결론 내리기 어렵다고 생각한다. 즉, 결론은 0.6과 0.65사이에서 가장 작은 average loss를 가질 것이라는 것이다.

- Comparation for accuracy

learning rate	accuracy

0.001	11.4
0.01	38.61
0.03	30
0.1	90.4
0.2	93.4
0.3	94.6
0.4	95.2
0.5	95.4
0.6	95.8
0.61	95.6
0.62	95.6
0.63	95.6
0.65	95.6
0.7	95.8

위 실험 결과로 accuracy는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.4이상에서 accuracy가 느는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았지만, accuracy는 정수로 epoch에 대해 %로만 결과가 나오기 때문에 정확한 판단을 하기 어려운 지표라고 생각했다. 위 average loss의 결과와 함께 생각해보았을 때 relu function은 0.6과 0.7사이의 learning rate에서 가장 좋은 성능을 낸다고 평가할 수 있다.

- Comparation for total training and evaluation time

learning rate	time
0.001	139.6413
0.01	107.9851
0.03	89.7824
0.1	107.7702
0.2	85.9246
0.3	106.0889
0.4	107.8933
0.5	108.027
0.6	107.7854
0.61	112.6515
0.62	95.8231

0.63	98.8849
0.65	94.7526
0.7	102.3338

위 실험 결과로 total time은 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. Accuracy와 average loss는 거의 반비례 관계를 가졌지만 time은 이 둘과 반비례 혹은 비례관계를 찾기 어려웠다.

앞서 최적의 성능을 내는 learning rate로 예상했던 0.6과 0.7사이구간에서 90초후반 혹은 100초대인 다른 rate와 다르게 0.65는 90초반대의 total time을 가진다.

앞선 세 지표로 인한 결과는 0.65~0.66사이의 learning rate일 때 sigmoid함수는 최적의 성능을 낼 수 있다는 것이다.

d. LeakyRelu with c=0.01

- Comparation for average loss

learning rate	average loss
0.001	1.3964
0.01	0.287
0.03	0.8
0.1	0.1232
0.15	0.118
0.16	0.12
0.17	0.118
0.18	0.1204
0.2	0.1242
0.25	0.1177
0.3	0.1213

위 실험 결과로 average loss는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.1~0.3사이에서 average loss가 크게 주는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았을 때 0.17~0.18 혹은 0.25에서 가장 작은 average loss를 가짐을 확인했다. 하지만 average loss는 같은 값을 실험하더라도 달라지기에 이를 섣불리 결론 내리기 어렵다고 생각한다. 즉, 결론은 0.1과 0.26사이에서 가장 작은 average loss를 가질 것이라는 것이다.

- Comparation for accuracy

learning rate	accuracy
0.001	66.8
0.01	91.6
0.03	86.6
0.1	96.2
0.15	96.2
0.16	96
0.17	96.2
0.18	96.2
0.2	96
0.25	96.2
0.3	96.2

위 실험 결과로 accuracy는 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. 또한 0.1이상에서 accuracy가 느는 것을 확인해서 소수 둘째점 사이까지 변경하여 실험해 보았지만, accuracy는 소수점자리 없이 각 epoch에 대해 %로만 결과가 나오기 때문에 정확한 판단을 하기 어려운 지표라고 생각했다. 위 average loss의 결과와 함께 생각해보았을 때 relu function은 0.1과 0.1-0.26사이의 learning rate에서 가장 좋은 성능을 낸다고 평가할 수 있다.

- Comparation for total training and evaluation time

learning rate	time
0.001	105.92
0.01	96.9142
0.03	89.1866
0.1	105.126
0.15	114.02
0.16	114.5356
0.17	106.44
0.18	104.6127
0.2	112.93
0.25	108.374

0.3	104.93

위 실험 결과로 total time은 learning rate와 비례 혹은 반비례 관계를 가지지 않음을 확인했다. Accuracy와 average loss는 거의 반비례 관계를 가졌지만 time은 이 둘과 반비례 혹은 비례관계를 찾기 어려웠다.

앞서 최적의 성능을 내는 learning rate로 예상했던 0.1과 0.25사이구간에서 0.18이 가장 작은 total time을 가진다.

앞선 세 지표로 인한 결과는 약 0.18의 learning rate일 때 leaky_relu함수는 최적의 성능을 낼 수 있다는 것이다

정리하자면 최적의 성능을 내는 learning rate가 relu function은 0.14~0.15, tanh function은 0.18~0.19, sigmoid function은 0.65~0.66, leaky_relu function은 0.18정도 이였다. 실험 결과로, sigmoid function은 다른 세 개의 함수에 비해 learning rate가 상당히 높았다. 즉, sigmoid function은 gradient descent algorithm에서 한번 경사를 내려갈 때 크게 이동해야 한다. 이 실험 결과로 learning rate은 실험 환경에 따라 적절히 설정해 주어야 할 필요성을 알 수 있었다.

3) Analysis-depth of hidden layers for each activation function

기존의 (784,50)->(50,50)->(50,10)의 형태를 띄는 layer를 다음과 같은 형태를 띄도록 __init__함수와 forward함수를 수정한 후 결과를 비교했다.

```
# Define the first fully connected
self.fc1 = nn.Linear(28 * 28, 50)

# Define the second fully connecte
self.fc2 = nn.Linear(50, 50)

self.fc3=nn.Linear(50,50)

# Define the third fully connected
self.fc4 = nn.Linear(50, 10)
```

(784,50)->(50,50)->(50,50)->(50,10)

a. Relu

-epoch 1

```
Test set: Average loss: 1.8194, Accuracy: 5615/10000 (56%)
```

-epoch 2

```
Test set: Average loss: 0.8905, Accuracy: 7488/10000 (75%)
```

-epoch 3

```
Test set: Average loss: 0.6241, Accuracy: 8194/10000 (82%)
```

-epoch 4

```
Test set: Average loss: 0.5300, Accuracy: 8478/10000 (85%)
```

-epoch 5

```
Test set: Average loss: 0.4864, Accuracy: 8611/10000 (86%)
```

-total training and evaluation time

```
Total training and evaluation time: 105.1558 초
```

1)에서 analysis한 relu function의 실험결과와 비교했을 때 average loss가 0.1~1.0정도 증가했음을 알 수 있다. 또한 accuracy는 10~20%정도 감소했으며 total time 또한 20초 정도 증가했다.

b. Tanh

-epoch 1

```
Test set: Average loss: 1.3725, Accuracy: 6753/10000 (68%)
```

-epoch 2

```
Test set: Average loss: 0.9493, Accuracy: 7806/10000 (78%)
```

-epoch 3

```
Test set: Average loss: 0.7661, Accuracy: 8355/10000 (84%)
```

-epoch 4

```
Test set: Average loss: 0.6736, Accuracy: 8542/10000 (85%)
```

-epoch 5

```
Test set: Average loss: 0.6219, Accuracy: 8634/10000 (86%)
```

-total training and evaluation time

```
Total training and evaluation time: 101.8948 초
```

1)에서 analysis한 tanh function의 실험결과와 비교했을 때 average loss가 0.2정도 증가했음을 알 수 있다. 또한 accuracy는 10%정도 감소했으며 total time은 거의 변화가 없다.

c. Sigmoid

-epoch 1

```
Test set: Average loss: 2.3005, Accuracy: 1135/10000 (11%)
```

-epoch 2

```
Test set: Average loss: 2.3003, Accuracy: 1135/10000 (11%)
```
S

-epoch 3

```
Test set: Average loss: 2.3001, Accuracy: 1135/10000 (11%)
```

-epoch 4

```
Test set: Average loss: 2.3000, Accuracy: 1135/10000 (11%)
```

-epoch 5

```
Test set: Average loss: 2.2999, Accuracy: 1135/10000 (11%)
```

-total training and evaluation time

```
Total training and evaluation time: 124.9037 초
```

1)에서 analysis한 sigmoid function의 실험결과와 비교했을 때 average loss는 거의 변화가 없고 accuracy는 6%정도 감소했음을 확인할 수 있다. Total time은 30초정도 증가한 것을 확인할 수 있다.

d. LeakyRelu with c=0.01

-epoch 1

```
Test set: Average loss: 1.8118, Accuracy: 5647/10000 (56%)
```

-epoch 2

```
Test set: Average loss: 0.8804, Accuracy: 7507/10000 (75%)
```

-epoch 3

```
Test set: Average loss: 0.6188, Accuracy: 8212/10000 (82%)
```

-epoch 4

```
Test set: Average loss: 0.5267, Accuracy: 8486/10000 (85%)
```

-epoch 5

```
Test set: Average loss: 0.4840, Accuracy: 8616/10000 (86%)
```
S

-total training and evaluation time

```
Total training and evaluation time: 110.8417 초
```

1)에서 analysis한 leaky relu function의 실험결과와 비교했을 때 average loss가 0.1~1.0정도 증가했음을 알 수 있다. 또한 accuracy는 10~20%정도 감소했으며 total time 또한 20초 정도 증가했다. relu함수의 depth를 변화시켰을 때와 거의 같은 결과를 얻었다.

위 실험 결과로 hidden layer의 depth를 증가시켰을 때, 모든 activation function에서 성능이 안 좋아졌음을 확인했다. 1.배경지식에서 sigmoid function은 hidden layer가 deep해질수록 마지막 layer에 거의 영향을 끼치지 못함을 언급했듯 가장 성능차이가 큰 것을 확인했다. 깊이를 더 깊게 만들수록 다른 function들에 비해 sigmoid function은 더 크게 성능이 안 좋아질 것으로 예상된다.

4) Analysis-width of hidden layer for each activation function

1. Reduce the width

   Original : (784,50)->(50,50)->(50,10)

   Changed thing : (784,10)->(10,10)->(10,10)

```
self.fc1 = nn.Linear(28 * 28, 10)

# Define the second fully connected
self.fc2 = nn.Linear(10, 10)

# Define the third fully connected l
self.fc3 = nn.Linear(10, 10)
```

   a. Relu

      -epoch 1

```
Test set: Average loss: 1.6806, Accuracy: 3932/10000 (39%)
```

-epoch 2

```
Test set: Average loss: 1.2112, Accuracy: 6150/10000 (62%)
```

-epoch 3

```
Test set: Average loss: 0.9521, Accuracy: 7096/10000 (71%)
```

-epoch 4

```
Test set: Average loss: 0.8305, Accuracy: 7465/10000 (75%)
```

-epoch 5

```
Test set: Average loss: 0.7646, Accuracy: 7662/10000 (77%)
```

-total training and evaluation time

```
Total training and evaluation time: 205.2022 초
```

1)에서 analysis한 relu function의 실험결과와 비교했을 때 average loss는 0.4~0.8정도 증가했으며 accuracy 또한 10~30%정도 감소했다. Toal time또한 120초 정도 증가했다.

b. Tanh

-epoch 1

```
Test set: Average loss: 1.6398, Accuracy: 5668/10000 (57%)
```

-epoch 2

```
Test set: Average loss: 1.4162, Accuracy: 7094/10000 (71%)
```

-epoch 3

```
Test set: Average loss: 1.2806, Accuracy: 7674/10000 (77%)
```

-epoch 4

```
Test set: Average loss: 1.1933, Accuracy: 7918/10000 (79%)
```

-epoch 5

```
Test set: Average loss: 1.1357, Accuracy: 8064/10000 (81%)
```

-total training and evaluation time

```
Total training and evaluation time: 94.0728 초
```

1)에서 analysis한 tanh function의 실험결과와 비교했을 때 average loss는 0.6 정도 증가했으며 accuracy는 10~20%감소했다. Total time은 4초정도 감소한 것으로 보아 거의 변화가 없다.

c. Sigmoid

-epoch 1

```
Test set: Average loss: 2.2939, Accuracy: 1135/10000 (11%)
```

-epoch 2

```
Test set: Average loss: 2.2771, Accuracy: 1135/10000 (11%)
```

-epoch 3

```
Test set: Average loss: 2.2676, Accuracy: 1467/10000 (15%)
```

-epoch 4

```
Test set: Average loss: 2.2612, Accuracy: 1781/10000 (18%)
```

-epoch 5

```
Test set: Average loss: 2.2567, Accuracy: 1908/10000 (19%)
```
S

-total training and evaluation time

```
Total training and evaluation time: 98.6371 초
```

1)에서 analysis한 sigmoid function의 실험결과와 비교했을 때 average loss와 accuracy는 약간의 증가와 감소는 했지만 다른 함수들의 결과들에 비하면 굉장히 미미한 변화이다. 또한 total time은 10초 정도 증가했다.

d. LeakyRelu with c=0.01

-epoch 1

```
Test set: Average loss: 1.6675, Accuracy: 4150/10000 (42%)
```

-epoch 2

```
Test set: Average loss: 1.2011, Accuracy: 6180/10000 (62%)
```

-epoch 3

```
Test set: Average loss: 0.9453, Accuracy: 7128/10000 (71%)
```

-epoch 4

```
Test set: Average loss: 0.8255, Accuracy: 7476/10000 (75%)
```

-epoch 5

```
Test set: Average loss: 0.7607, Accuracy: 7678/10000 (77%)
```

-total training and evaluation time

```
Total training and evaluation time: 224.4302 초
```

1)에서 analysis한 leaky relu function의 실험결과와 비교했을 때 average loss는 0.4~0.8정도 증가했으며 accuracy는 10~40%감소했다. Total time은 130초 정도 증가했다.

중간 hidden layer의 width를 감소시켰을 때 모든 함수가 성능이 나빠졌다 판단할 수 있지만, sigmoid function에서는 변화가 거의 없었다.

2. Increase the width

Original : (784,50)->(50,50)->(50,10)

Changed thing : (784,150)->(150,150)->(150,10)

```
# Define the first fully connected
self.fc1 = nn.Linear(28 * 28, 150)

# Define the second fully connected
self.fc2 = nn.Linear(150, 150)

# Define the third fully connected
self.fc3 = nn.Linear(150, 10)
```

a. Relu

-epoch 1

```
Test set: Average loss: 0.7801, Accuracy: 8200/10000 (82%)
```

-epoch 2

```
Test set: Average loss: 0.4820, Accuracy: 8737/10000 (87%)
```

-epoch 3

```
Test set: Average loss: 0.4099, Accuracy: 8903/10000 (89%)
```

-epoch 4

```
Test set: Average loss: 0.3804, Accuracy: 8948/10000 (89%)
```

-epoch 5

```
Test set: Average loss: 0.3650, Accuracy: 8974/10000 (90%)
```

-total training and evaluation time

```
Total training and evaluation time: 120.0519 초
```

1)에서 analysis한 relu function의 실험결과와 비교했을 때 average loss는 약간 감소하였고 accuracy는 epoch1에서 2%증가했고 다른 epoch에서는 거의 비슷했다. Total tie 은 35초 정도 증가했다.

b. Tanh

-epoch 1

```
Test set: Average loss: 0.7730, Accuracy: 8272/10000 (83%)
```

-epoch 2

```
Test set: Average loss: 0.4817, Accuracy: 8787/10000 (88%)
```

-epoch 3

```
Test set: Average loss: 0.4080, Accuracy: 8917/10000 (89%)
```

-epoch 4

```
Test set: Average loss: 0.3768, Accuracy: 8997/10000 (90%)
```

-epoch 5

```
Test set: Average loss: 0.3605, Accuracy: 9025/10000 (90%)
```

-total training and evaluation time

```
Total training and evaluation time: 119.2719 초
```

1)에서 analysis한 tanh function의 실험결과와 비교했을 때 average loss는 0.2~0.7정도 감소하였고 accuracy도 약간이지만 전체적으로 증가한 것을 확인할 수 있다. Total time은 20초정도 증가하였다.

c. Sigmoid

-epoch 1

```
Test set: Average loss: 2.2848, Accuracy: 2118/10000 (21%)
```

-epoch 2

```
Test set: Average loss: 2.2716, Accuracy: 2975/10000 (30%)
```

-epoch 3

```
Test set: Average loss: 2.2604, Accuracy: 3453/10000 (35%)
```

-epoch 4

```
Test set: Average loss: 2.2514, Accuracy: 3575/10000 (36%)
```

-epoch 5

```
Test set: Average loss: 2.2442, Accuracy: 3597/10000 (36%)
```

-total training and evaluation time

```
Total training and evaluation time: 165.2818 초
```

1)에서 analysis한 sigmoid function의 실험결과와 비교했을 때 average loss는 거의 변화가 없으며 accuracy는 10%~20%정도 증가했다. Total time은 75초 증가하였다.

d. LeakyRelu with c=0.01

-epoch 1

```
Test set: Average loss: 0.7777, Accuracy: 8205/10000 (82%)
```

-epoch 2

```
Test set: Average loss: 0.4812, Accuracy: 8741/10000 (87%)
```

-epoch 3

```
Test set: Average loss: 0.4095, Accuracy: 8905/10000 (89%)
```

-epoch 4

```
Test set: Average loss: 0.3802, Accuracy: 8950/10000 (90%)
```

-epoch 5

```
Test set: Average loss: 0.3649, Accuracy: 8981/10000 (90%)
```

-total training and evaluation time

```
Total training and evaluation time: 114.0917 초
```

1)에서 analysis한 leaky relu function의 실험결과와 비교했을 때 average loss와 accuracy 모두 미미하게 감소하고 증가했다. Total time은 20초정도 증가했다.

중간 hidden layer의 width를 증가시켰을 때, 모든 함수는 average loss와 accuracy면에서 성능이 좋아졌다 판단할 수 있지만 total time에선 상당한 손해가 있었다.

위 실험 결과로 hidden layer의 width또한 average loss와 accuracy 그리고 total time면에서 상당한 영향을 미침을 알 수 있었다. 그러므로 우리는 주어진 데이터와 시간에 따라 올바른 layer의 width와 function을 선택해야 한다.