인공지능(딥러닝)개론 HW1 결과보고서

학번: 20180032 이름: 남기동

1. 2-layer 모델

```
net1 = sigmoid(np.matmul(xin,W1)+b1)
net2 = sigmoid(np.matmul(net1,W2)+b2)
```

입력 Xin과 W1, b1을 통해 hidden layer의 출력인 net1을 계산하고 이를 바탕으로 W2, b2를 통해 output layer의 출력인 net2를 계산한다. 이 때 Activation Fuction으로 sigmoid를 사용한다.

```
loss = np.mean((net2 - ans) ** 2)
```

output layer의 출력과 실제값 사이의 차이, 즉 오차를 제곱하고 평균을 내어 loss를 계산한다. 이는 MSE(Mean Squared Error)로 loss를 구한 것이다

```
def sigmoid_derivative(x):
    return x * (1 - x)
```

sigmoid 함수의 경우 미분값은 x(1-x) 꼴이라는 것을 수업 시간에 증명하였고 이를 활용한다.

```
delta_output = 2 * (net2 - ans) * sigmoid_derivative(net2)
delta_W2 = np.dot(net1.T, delta_output)
delta_b2 = np.sum(delta_output)
```

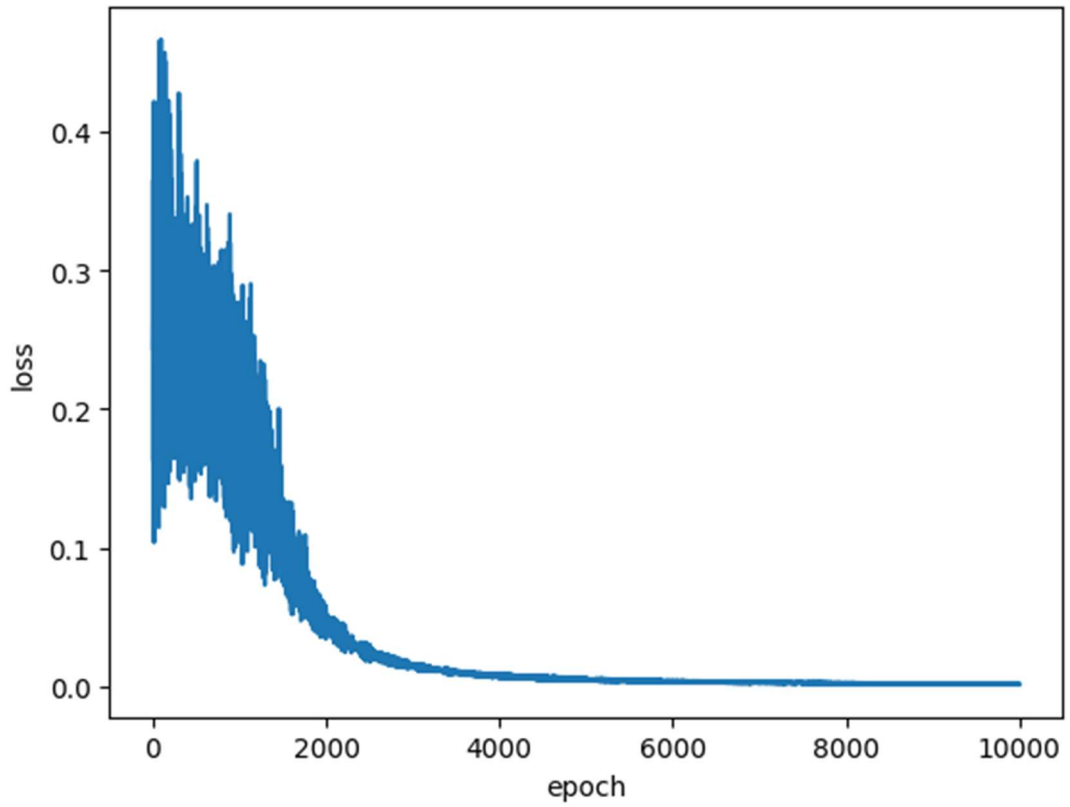
Backpropagation을 통해 weight의 gradient를 계산하는 부분이다. 우선 output layer에서 delta_output은 Loss를 W2에 대해 미분한 값이다. Loss가 MSE이기 때문에 제곱의 2가 앞으로 나가 2*(net2-ans)가 되고 속미분을 해야 하는데, sigmoid 함수를 activation function으로 하기 때문에 미분값이 net2 * (1 – net2)이 되어 곱해진다 이 값을 hidden layer의 출력값인 net1과 내적한 값이 W2를 업데이트 하기 위한 값이 된다.

```
delta_hidden = np.dot(delta_output, W2.T) * sigmoid_derivative(net1)
delta_W1 = np.dot(xin.T, delta_hidden)
delta_b1 = np.sum(delta_hidden)
```

output layer에서 했던 방식과 동일하게 hidden layer에서 수행하고 있다.

```
W1 = W1 - learning_rate * delta_W1
W2 = W2 - learning_rate * delta_W2
b1 = b1 - learning_rate * delta_b1
b2 = b2 - learning_rate * delta_b2
```

learning_rate을 곱해 step size를 조절하고 -1을 곱해서 gradient의 반대 방향의 값으로 기존 값에 더해서 minimum point로 갈 수 있도록 한다.

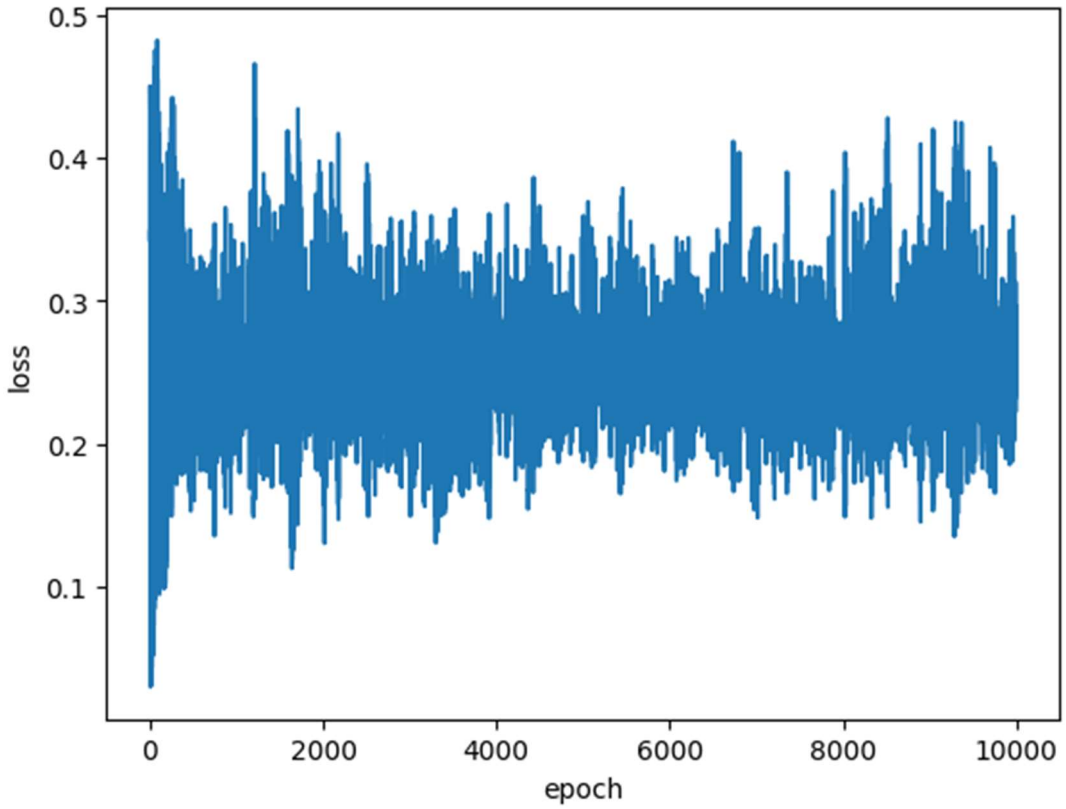



```
input:  [0 0] , answer:  0 , pred: 0.0450
input:  [0 1] , answer:  1 , pred: 0.9585
input:  [1 0] , answer:  1 , pred: 0.9632
input:  [1 1] , answer:  0 , pred: 0.0427
```

이는 2-layer 모델에서 모델을 학습한 횟수에 따라 loss가 감소한 것을 그래프로 나타낸 것이다. 처음에는 큰 step으로 변화하면서 loss가 0에 가까워지는 방향으로 감소하고 점점 loss가 감소함에 따라 변동폭이 감소하는 것을 확인할 수 있다. 결과적으로 각각의 input에 대해 정답이 매우 적은 오차로 출력되는 것을 확인할 수 있다.

2. 1-layer 모델

1-layer 모델은 2-layer 모델에서 가졌던 hidden layer이 없이 input에서 바로 output으로 가는 구조를 갖고 있다. 코드의 기본 구조는 2-layer 모델과 동일하고 net2를 사용하지 않을 뿐이기 때문에 자세한 설명은 생략한다.




```
input:  [0 0] , answer:  0 , pred: 0.4696
input:  [0 1] , answer:  1 , pred: 0.4971
input:  [1 0] , answer:  1 , pred: 0.4813
input:  [1 1] , answer:  0 , pred: 0.5089
```

1-layer 모델의 학습 횟수에 따른 loss의 그래프를 살펴보면 2-layer 모델과 매우 다르다는 것을 확인할 수 있다. 2-layer 모델은 첫 번째 layer에서의 입력을 비선형 공간으로 매핑하고 두 번째 layer에서 이를 선형 결합하여 출력을 생성하여 선형으로 분리할 수 없는 XOR 문제를 해결했다. 그러나 1-layer 모델은 이러한 작업을 하지 못하여 결정 경계가 직선으로 표현되는, 입력 공간이 두 영역으로 나뉘는 결정 경계가 직선으로 표현될 수 있는 문제만 해결할 수 있다. 따라서 학습 횟수가 늘어나도 XOR 문제를 해결하지 못하고 오차가 일정 값 이하로 떨어지지 못하고 있다.