4장. 다층 퍼셉트론 기본 구조

1. 개요

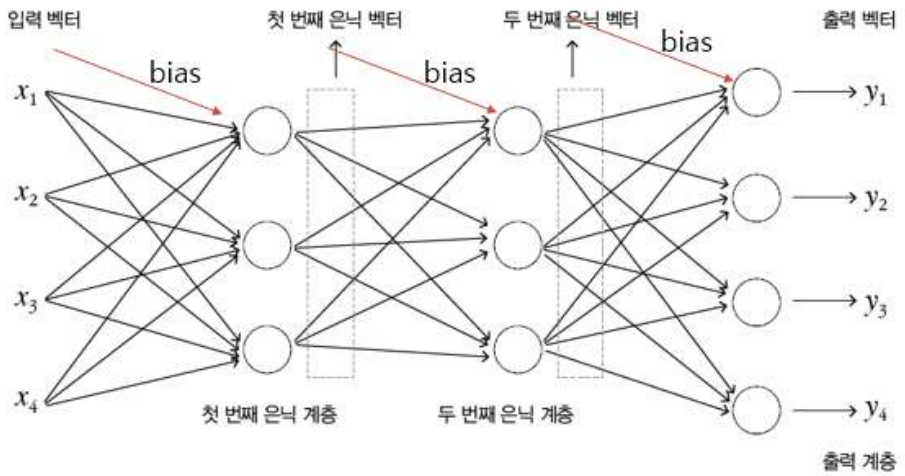
이번 과제는 다층 퍼셉트론 신경망을 통해 이전의 문제들을 해결하는 것이다. "전복 고리 수 추정", "천체 펄서 판정", "철판 불량 유형 분류" 3가지 문제를 다층 퍼셉트론 신경망을 통해 해결하고, 하이퍼파라미터 및 은닉계층의 수와 폭을 변경해가며 실험을 진행할 것이다. 또한, 이러한 변화가 다층 퍼셉트론 신경망에 어떠한 영향을 미치는지에 대해 살펴보고자 한다.

2. 이론

< 다층 퍼셉트론 신경망 - MultiLayer Perceptron >

단층 퍼셉트론의 한계를 개선하기 위해 이후에 나온 것이 다층 퍼셉트론 신경망이다. 단층 퍼셉트론과 달리 다층 퍼셉트론의 경우 은닉 계층(hidden layer)이 존재 하며, 복수의 퍼셉트론 계층을 순서를 두고 배치하여 입력 벡터로부터 중간 표현을 거쳐 출력 벡터를 얻어낸다. 또한 입력의 경우 편향(bias)도 항상 입력 되어야 한다.

그림 다층 퍼셉트론의 구성 예



< 출처 : 파이썬 날코딩으로 알고 짜는 딥러닝 >

위의 그림에서 볼 수 있듯이, 출력에 직접 나타나지 않는 계층을 은닉 계층이라고 하며, 은닉 계층이 생성하는 중간 표현을 은닉 벡터라고도 한다. 또한, 다층 퍼셉트론에서 각각의 계층은 단층 퍼셉트론과 같은 내부 구조를 갖는다. 이를 자세히 보면 하나의 계층 안에 속한 퍼셉트론들은 동일한 입력을 공유하면서 상호간의 어떠한 연결도 없이 영향을 주고받지 않고 출력 성분을 만들어내는 것을 알 수 있다. 반대로 인접한 계층 간에는 앞의 계층의 출력이 뒤의 계층의 모든 퍼셉트론에 공통 입력으로 들어가기 때문에, 인접 계층 간에는 방향성을 갖는 완전 연결 방식(fully connected)으로 연결되는 것을 확인 할 수 있다.

다층 퍼셉트론 신경망은 단층 퍼셉트론 구조에 비해 더 많은 퍼셉트론을 이용해 기억 용량이나 계산 능력에 대한 부담이 커지는 반면, 품질 향상을 기대할 수 있는 장점이 존재한다.

< 은닉 계층의 수와 폭 >

다층 퍼셉트론에서 최종 단계에 배치된 계층의 경우 신경망에 주어진 원래의 임무에 따라 알맞은 형태의 출력 벡터를 생성하는 역할을 하고 있다. 그에 따라 출력 계층이라는 별도의 이름을 가지며 출력 계층이 가질 퍼셉트론 수도 문제의 성격에 따라 고정적으로 정해진다. 반면에 은닉 계층이 생성하는 은닉 벡터의 경우 이러한 제약이 없기 때문에 은닉 계층의 수와 각각의 폭(Width)은 신경망 설계자가 자유롭게 설정할 수 있다. 그러나 주의해야할 점이 있다. 은닉 계층의 수와 각각의 은닉 계층에 대한 폭은 신경망의 품질에 영향을 끼지는 중요한 요인이 될 수 있지만, 무조건 은닉 계층 수나 폭을 늘린다고 품질이 좋아지는 것은 아니다. 은닉 계층을 추가하여 파라미터 수가 늘어나면 그에 따라

필요한 학습 데이터의 양도 늘어나게 되는데 만약, 충분한 양의 적절한 데이터가 없으면 오버피팅(Overfitting)의 발생 가능성이 높아져 다층 퍼셉트론 구조의 확장이 오히려 신경망 능력을 떨어지게 할 수도 있다.

따라서, 은닉 계층의 수와 폭은 우리가 해결하고자 하는 문제의 크기, 성격, 준비된 데이터의 양, 난이도를 종합적으로 고려해서 정해야 한다. 그렇기에 다양한 실험과 축척된 경험이 중요하게 작용하며, 학습률, 미니배치 크기 같은 다른 하이퍼 파라미터도 중요한 영향을 미칠 수 있으므로 조심해야 한다. 따라서 다층 퍼셉트론을 문제 해결에 도입할 경우 이러한 점으로 인해 은닉 계층 수와 폭의 설정 값을 쉽게 바꾸어가며 실험할 수 있도록 프로그램을 구현해야 한다.

< 오버피팅(Overfitting) >
학습 데이터를 과하게 학습(Overfitting)하는 것을 뜻한다. 일반적으로 학습 데이터는 실제 데이터의 부분 집합이므로 학습 데이터에 대해서는 오차가 감소하고 잘 작동하지만, 실제 데이터에 대해서는 오차가 증가하는 것을 뜻한다. 이는 모델의 성능을 떨어트리는 주요 문제점이며, 이를 해결하기 위한 여러 가지 방법이 존재한다.

< 선형 연산 >
계수를 곱하는 곱셈과 덧셈으로만 이루어지는 일차 연산을 뜻하며, 활성화 함수로 선형 함수 사용시 아무리 많은 레이어를 사용하여도 궁극적으로는 선형이 된다.

< 비선형 활성화 함수 >
은닉 계층은 가중치와 편향을 이용해 계산된 선형 연산 결과를 바로 출력으로 내보내지 않고 한 번 더 변형시켜 내보낸다. 이 때, 선형 연산 결과 뒷단에 적용하여 퍼셉트론의 출력을 변형시키기 위해 추가한 것을 비선형 활성화 함수라고 한다. 비선형 함수는 일차 함수로 표현이 불가능한 좀 더 복잡한 기능을 수행하는 함수이기에 우리는 비선형 활성화 함수를 추가함으로써 입력의 일차 함수 표현을 뛰어넘어 다양하고 복잡한 형태의 퍼셉트론 출력을 만들 수 있다.

우리는 이전의 과제들을 통해서 시그모이드 함수, 소프트맥스 함수와 같은 대표적인 비선형 함수들을 만나보았다. 이전의 단층 퍼셉트론에서는 이들을 퍼셉트론 내의 구성 요소로 간주하지 않았으며, 이는 출력 계층의 출력 유형에 따라 후처리 과정에서 비선형 함수를 이용하는 방법이 달라지고 비선형 함수를 퍼셉트론 안에 두기가 곤란하기 때문이다. 예를 들어 소프트 맥스 함수의 경우를 보자. 소프트 맥스 함수의 경우 출력 여러 개를 묶어 함께 처리해야 하며 따라서 각기 독립된 구조인 개별 퍼셉트론 안에는 이 함수를 나누어서 넣을 수가 없다.

또한, 출력 벡터 1개 안에 각기 다른 역할을 하는 여러 정보를 함께 생성해야 할 경우도 있다. 이러한 경우 출력 벡터를 구성하는 각 정보의 특성에 맞게 별도의 비선형 처리를 해야 한다. 그러나 이를 위해 출력 계층 안에 여러 가지 비선형 활성화 함수를 넣게 되면 코드가 지나치게 복잡해지므로 출력 계층에 비선형 활성화 함수를 두지 않는다. 이러한 이유로 출력 계층에는 비선형 활성화 함수가 사용되지 않으며, 단층 퍼셉트론에서는 후처리 과정에서 시그모이드 관련 함수나 소프트맥스 관련 함수를 이용하였을 뿐이다.

위의 내용을 정리하면 다음과 같다.
< 출력 계층에는 비선형 활성화 함수 사용 X >
- 시그모이드 함수나 소프트맥스 함수 이용한 후처리로 대신한다.
- 시그모이드, 소프트맥스 2개의 함수 모두 비선형 함수이지만 출력 계층에는 부적절하다.
- 소프트맥스 함수의 경우 다대다 함수이므로 퍼셉트론 내에 삽입이 힘들다.
- 출력 성분 별로 서로 다른 후처리 과정이 필요하다.(복합 출력)

출력계층과는 달리 은닉 계층은 비선형 활성화 함수가 선형 연산 결과를 변형시켜 퍼셉트론의 최종 출력을 만들어 낸다. 은닉 계층의 다음 계층에는 또 다른 은닉 계층이나 출력 계층이 존재하기 때문에 다층 퍼셉트론에서는 계층과 계층 사이의 선형 연산 사이마다 비선형 활성화 함수가 놓이게 된다. 선형처리를 아무리 반복하여도 하나의 선형 처리로 표현할 수 있다는 수학적 원리로 인해서, 다층 퍼셉트론에서 비선형 활성화 함수는 필수적 요소이다.

이해를 돕기 위해 다중 선형 연산에 대해서 다음과 같이 정리 하였다.

[다중 선형 연산]

- 다중 선형 연산의 불필요성에 대해서 다음과 같은 예시를 통해 확인할 수 있다.
- 다중 선형 연산은 한 층의 선형 연산으로 대치가 가능하다.

예시 1) y = 2x + 3, z = 3y − 4 ⇒ z = 6x + 5

예시 2) 2개의 층의 선형 퍼셉트론이 차례로 처리하는 경우

- 층1의 가중치가 w, 편향이 b일 때 : $h_i = w_{i1}x_1 + \cdots + w_{\in}x_n + b \quad \cdots\cdots (1)$
- 층2의 가중치가 v, 편향이 c일 때 : $y_j = v_{j1}h_1 + \cdots + v_{jm}h_m + c \quad \cdots\cdots (2)$
- (2) 에 (1)을 대입 :

$$y_j = (w_{11}v_{j1} + \cdots + w_{m1}v_{jm})x_1 + \cdots + (w_{1n}v_{j1} + \cdots + w_{mn}v_{jm})x_n + ((v_{j1} + \cdots + v_{jm})b + c) \cdots\cdots (3)$$

- w, v, b, c 가 결정될 경우 값을 대입해 (3) 식의 계수 및 상수 값을 결정 가능하다.
  => 즉, 2 층의 선형 연산을 1개의 층으로 대치 가능하다.

그림 카이사르 암호법에 따른 중복 암호화의 불필요성

< 다중 선형 연산의 불필요성 이해를 돕기 위한 그림 >

우리는 위에서 다중 선형 연산에 대해서 알아보았다. 이러한 선형성의 한계를 극복하기 위해서 우리는 비선형 함수를 사용하는 것이며, 적당히 비선형 함수 A를 도입하여 $h_1 = A(w_{i1}x_1 + \ldots + w_{\in}x_n + b)$ 로 변형하게 되면 선형성의 한계에서 벗어날 수 있다. 만약 적절한 수의 퍼셉트론으로 구성되어 있고, 비선형 활성화 함수를 갖춘 은닉 계층에 우리가 가중치, 편향값을 잘 설정해주면 단 2계층의 다층 퍼셉트론만으로 어떤 수학적 함수이든 원하는 오차 수준 이내로 근사하게 동작하게 할 수 있을 것이다. 이는 수학적으로도 증명되었다고 한다.

다양한 실험에 따르면 노드 수가 많은 단층 구조의 신경망보다 노드 수가 적은 다층 구조의 신경망 성능이 훨씬 우수한 경우가 많다. 이는 입력 데이터로부터 유용한 정보를 추출해내는 추상화에 단층 구조보다 다층 구조가 효과적인 것으로 볼 수 있으며, 이러한 이유로 인해 계층 수는 적지만 전체적으로 많은 노드를 갖는 신경망보다 계층 수는 많아도 노드가 적은 신경망으로 문제를 해결하려는 경향이 나타나게 된다.

위의 내용을 PPT 자료를 통해 정리하면 다음과 같다.

< 비선형 활성화 함수의 활용에 대한 정리 >

- 비선형 활성화 함수란?
  - 비선형 함수 : 선형 연산 결과를 비선형으로 변형
  - 활성화 함수 : 은닉 계층 뒷단에 추가해 퍼셉트론 출력 변형
- 출력 계층
  - 출력 계층은 비선형 활성화 함수를 두지 않는다.
  - 시그모이드 함수나 소프트 맥스 함수를 이용한 후처리로 대신하며,
  - 두 함수 모두 비선형 함수지만 출력 계층에는 부적절하다.
  (소프트 맥스 함수를 은닉 계층에 사용하기는 매우 힘듬)
  - 소프트 맥스 함수의 경우 다대다 함수로 퍼셉트론 내부에 삽입이 힘들다.
  - 출력 성분 별로 서로 다른 후처리 과정이 필요하다. (복합 출력)
- 은닉 계층
  - 비선형 활성화 함수 사용이 필수적이다.
  - 비선형 활성화 함수 없이는 독립된 계층으로서의 의미가 없다.
  - 두 층의 선형처리는 1개 층으로 압축 가능하다.
  - 다양한 종류의 비선형 함수가 가능하지만 ReLU를 가장 널리 이용한다.

< 다중 퍼셉트론의 가능성에 대한 정리 >

- 다중 퍼셉트론 신경망 : 임의의 함수를 모사할 수 있다.
  - 1개의 은닉 계층과 1개의 출력 계층만으로 충분하며, 임의의 함수를 원하는 정밀도 내에서 근사 처리 가능하다. 이는 수학적으로 증명된 다층 퍼셉트론 신경망의 성질이며, 단 두 계층에는 충분한 수의 퍼셉트론 배치가 필요하다. (두 계층의 파라미터 값은 적절히 설정되어 있어야 함)
- 수학적 증명에 따른 신경망 구현의 문제점
  - 많은 문제들이 지나치게 많은 수의 퍼셉트론이 필요하다.
  - 적절한 파라미터 값 설정을 위한 방법론이 존재하지 않는다.
- 현실적 접근
  - 두 층에 많은 퍼셉트론보다 많은 층에 적은 수의 퍼셉트론이 효과적이다.
  - 딥러닝의 딥(Deep) 표현이 사용되는 이유에 해당한다.
  - 경사하강법 등을 이용한 근사적 합습
    -> 수학적 최적의 파라미터 값에 대한 학습을 포기하고,
    -> 경험적 최적의 파라미터 값에 대한 학습으로 대치한다.

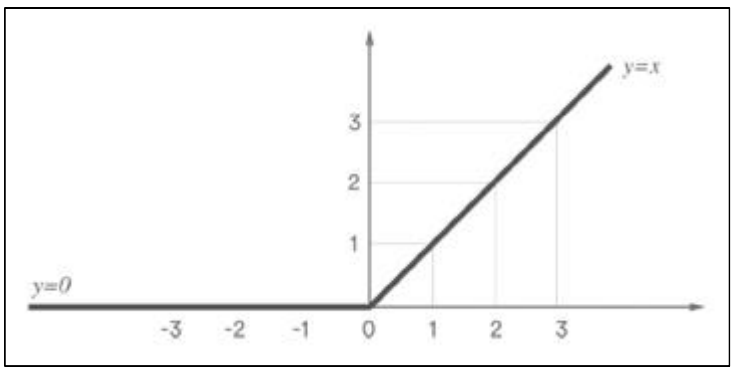
< ReLU 함수 - Rectified Linear Unit >

음수 입력을 걸러내 0으로 만드는 간단한 기능을 제공한다. 또한, 빠르게 계산할 수 있는 매우 값싼 함수이며 은닉 계층의 비선형 활성화 함수로 가장 널리 이용된다. 정의는 다음과 같다.

- 정의 : x > 0 일 때, y=x
  x <= 0 일 때, y=0

- ReLU함수의 그래프는 꺾인 선 모양으로 곡선 부분이 존재하지 않는다.

- ReLU함수의 그래프 곧은 직선 모양이 아니기 때문에 비선형 함수이다.



< ReLU 함수의 그래프 >

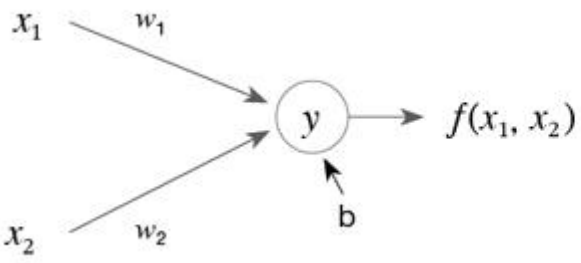
< ReLU 함수의 미분 - PPT 자료 활용 >

• 입력 값을 이용한 미분으로 x > 0 일 때 y' = 1, x <0 일 때, y′ = 0 이 된다.
(x = 0에서는 미분 불가능으로 x=0일 때 y′ = 0으로 정의해도 무방하다.)
• 출력 값을 이용한 미분으로 y > 0 일 때 y′ =1, y = 0일 때 y′ = 0 이 된다.
(y < 0 일 때 y′ = -1로 정의가 가능하다. -> 어차피 존재 불가능하므로 마음대로 정의한다.)
(np.sign(y) 함수를 이용하여 간편하게 계산 가능하다.)

< 민스키의 XOR 문제 >

우리는 첫 번째 과제에서 XOR 문제를 다루었으므로 간단하게 PPT 내용을 통해 XOR문제를 정리하였다.

[단층 퍼셉트론과 XOR 문제]



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

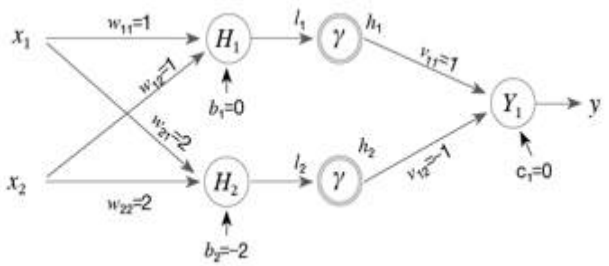
• $y = f(x1, x2) = w_1x_1 + w_2x_2 + b$

• x_2 와 y 사이의 관계
w_2 > 0 일 때 : x_2이 증가하면 y도 증가
w_2 = 0 일 때 : x_2의 변화는 y 값에 영향 X
w_2 < 0 일 때 : x_2가 증가하면 y는 감소

• w_1, w_2, b 값이 정해져 있는 상황에서는 노랑(x_2 커지면 y 증가), 녹색(x_2 커지면 y 감소) 동시에 만족이 불가능하며 입장을 바꾸어 x_1에 대해 생각해도 마찬가지이다.

민스키의 결론 : 퍼셉트론 으로는 XOR 기능조차 구현할 수 없다.

[다층 퍼셉트론과 XOR 문제]



x_1	x_2	l_1	l_2	h_1	h_2	y
0	0	0	-2	0	0	0
0	1	1	0	1	0	1
1	0	1	0	1	0	1
1	1	2	2	2	2	0

• 간단한 다층 퍼셉트론으로 XOR 구현 가능
– 출력 계층(1 퍼셉트론)에 은닉 계층(2 퍼셉트론) 추가
– 비선형 활성화 함수로는 ReLU 추가
– $w_{11}, w_{12}, w_{21}, w_{22}, b_1, b_2, v_{11}, v_{12}, c_1$의 값을 위 그림처럼 지정

3. 실험 결과 및 결론

3-1. < mlp_test.ipynb >

[전복 고리 수 추정 문제] 3-1

구분 (3.1)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 0 / 0
결과화면	은닉 계층 0개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=33.875, accuracy=0.557/0.812 Epoch 2: loss=8.226, accuracy=0.820/0.814 Epoch 3: loss=7.582, accuracy=0.812/0.809 Epoch 4: loss=7.475, accuracy=0.808/0.811 Epoch 5: loss=7.395, accuracy=0.810/0.809 Epoch 6: loss=7.328, accuracy=0.808/0.810 Epoch 7: loss=7.269, accuracy=0.808/0.811 Epoch 8: loss=7.217, accuracy=0.808/0.812 Epoch 9: loss=7.175, accuracy=0.810/0.810 Epoch 10: loss=7.135, accuracy=0.809/0.810 Final Test: final accuracy = 0.81
데이터 정확도	81.0%

※ 은닉 계층의 수가 0이므로 단층 퍼셉트론과 같은 성능을 내는 것을 확인할 수 있다.

구분 (3.2)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 1 / 4
결과화면	은닉 계층 하나를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=65.166, accuracy=0.299/0.564 Epoch 2: loss=11.775, accuracy=0.763/0.796 Epoch 3: loss=7.387, accuracy=0.794/0.800 Epoch 4: loss=7.291, accuracy=0.798/0.812 Epoch 5: loss=7.248, accuracy=0.801/0.803 Epoch 6: loss=7.212, accuracy=0.801/0.804 Epoch 7: loss=7.183, accuracy=0.802/0.809 Epoch 8: loss=7.143, accuracy=0.804/0.800 Epoch 9: loss=7.095, accuracy=0.803/0.810 Epoch 10: loss=7.057, accuracy=0.803/0.814 Final Test: final accuracy = 0.814
데이터 정확도	81.4%

구분 (3.3)	내용
조건	- 학습률(learning rate) : 0.09 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 50 - 은닉 계층 수 / 폭 : 1 / 4
결과화면	은닉 계층 하나를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 5: loss=10.777, accuracy=0.726/0.736 Epoch 10: loss=10.743, accuracy=0.728/0.688 Epoch 15: loss=10.765, accuracy=0.727/0.733 Epoch 20: loss=10.761, accuracy=0.727/0.732 Epoch 25: loss=10.768, accuracy=0.727/0.734 Epoch 30: loss=10.753, accuracy=0.727/0.727 Epoch 35: loss=10.782, accuracy=0.727/0.707 Epoch 40: loss=10.742, accuracy=0.728/0.700 Epoch 45: loss=10.807, accuracy=0.727/0.730 Epoch 50: loss=10.740, accuracy=0.729/0.706 Final Test: final accuracy = 0.706
데이터 정확도	70.6%

구분 (3.4)	내용
조건	- 학습률(learning rate) : 0.09 - 매개변수 인수값(mb_size) : 50 - 학습량(epoch) : 100 - 은닉 계층 수 / 폭 : 1 / 4
결과화면	은닉 계층 1개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 10: loss=6.734, accuracy=0.809/0.799 Epoch 20: loss=6.535, accuracy=0.808/0.728 Epoch 30: loss=6.188, accuracy=0.815/0.854 Epoch 40: loss=6.195, accuracy=0.807/0.850 Epoch 50: loss=6.212, accuracy=0.809/0.852 Epoch 60: loss=6.159, accuracy=0.812/0.794 Epoch 70: loss=6.332, accuracy=0.804/0.846 Epoch 80: loss=6.205, accuracy=0.809/0.847 Epoch 90: loss=6.500, accuracy=0.804/0.797 Epoch 100: loss=5.748, accuracy=0.823/0.768 Final Test: final accuracy = 0.768
데이터 정확도	76.8%

구분 (3.5)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 2 / [4, 4]
결과화면	은닉 계층 2개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=65.043, accuracy=0.298/0.525 Epoch 2: loss=22.413, accuracy=0.670/0.754 Epoch 3: loss=10.773, accuracy=0.740/0.737 Epoch 4: loss=10.041, accuracy=0.735/0.745 Epoch 5: loss=8.746, accuracy=0.750/0.761 Epoch 6: loss=7.820, accuracy=0.769/0.779 Epoch 7: loss=7.434, accuracy=0.781/0.795 Epoch 8: loss=7.259, accuracy=0.789/0.790 Epoch 9: loss=7.149, accuracy=0.793/0.795 Epoch 10: loss=7.077, accuracy=0.796/0.801 Final Test: final accuracy = 0.801
데이터 정확도	80.1%

구분 (3.6)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 4 / [2, 2, 2, 2]
결과화면	은닉 계층 4개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=65.212, accuracy=0.295/0.534 Epoch 2: loss=24.489, accuracy=0.648/0.739 Epoch 3: loss=12.402, accuracy=0.744/0.753 Epoch 4: loss=10.594, accuracy=0.732/0.742 Epoch 5: loss=10.571, accuracy=0.728/0.742 Epoch 6: loss=10.571, accuracy=0.728/0.741 Epoch 7: loss=10.576, accuracy=0.728/0.742 Epoch 8: loss=10.573, accuracy=0.728/0.742 Epoch 9: loss=10.572, accuracy=0.728/0.742 Epoch 10: loss=10.575, accuracy=0.728/0.742 Final Test: final accuracy = 0.742
데이터 정확도	74.2%

[전복 고리 수 추정 - 실험 결과]
모두 6번의 실험을 진행 하였으며, 1번째 실험의 경우 0개의 은닉 계층을 가지도록 설정 하였습니다. 따라서 이전의 과제와 동일한 결과를 볼 수 있었으며, 이 후의 실험에서는 1개의 은닉 계층에서 파라미터의 변화를 주었습니다. 그에 따라 최대 83.4% 까지 정확도가 증가하는 모습을 볼 수 있었으나, 70.6% 까지 감소하는 경우도 보았습니다.

이를 통해 늘어난 은닉층의 조건에서도 적절한 하이퍼 파라미터 값이 정확도에 얼마나 영향을 끼치는지 확인할 수 있었습니다. 마지막 2개 (3.5 , 3.6) 경우는 은닉 계층의 수와 폭에만 변화를 주었습니다. 은닉 계층이 1개 (폭:4 - 3.2)일 경우에는 기존의 결과보다 조금 상승한 모습을 볼 수 있었지만, 은닉 계층의 수와 폭을 점점 늘릴수록 정확도는 74.2% 까지 떨어지는 모습을 볼 수 있었습니다.

이를 통해 우리는 은닉 계층의 수 와 폭을 무작정 늘린다고 해서 신경망의 성능이 향상되지는 않는다는 것을 확인 할 수 있었습니다.

[천체 펄서 판정 문제] 3-2

구분 (3-2.1)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 0 / 0
결과화면	은닉 계층 0개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=0.154, accuracy=0.959/0.972 Epoch 2: loss=0.131, accuracy=0.966/0.972 Epoch 3: loss=0.136, accuracy=0.967/0.970 Epoch 4: loss=0.133, accuracy=0.968/0.970 Epoch 5: loss=0.121, accuracy=0.968/0.969 Epoch 6: loss=0.145, accuracy=0.968/0.974 Epoch 7: loss=0.122, accuracy=0.970/0.975 Epoch 8: loss=0.127, accuracy=0.970/0.976 Epoch 9: loss=0.125, accuracy=0.970/0.976 Epoch 10: loss=0.134, accuracy=0.968/0.976 Final Test: final accuracy = 0.976
데이터 정확도	97.6%

구분 (3-2.2)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 50 - 은닉 계층 수 / 폭 : 1 / 6
결과화면	은닉 계층 하나를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 5: loss=0.097, accuracy=0.970/0.969 Epoch 10: loss=0.094, accuracy=0.972/0.972 Epoch 15: loss=0.093, accuracy=0.972/0.974 Epoch 20: loss=0.091, accuracy=0.973/0.974 Epoch 25: loss=0.092, accuracy=0.973/0.970 Epoch 30: loss=0.091, accuracy=0.973/0.974 Epoch 35: loss=0.090, accuracy=0.974/0.975 Epoch 40: loss=0.089, accuracy=0.974/0.974 Epoch 45: loss=0.090, accuracy=0.974/0.974 Epoch 50: loss=0.089, accuracy=0.974/0.975 Final Test: final accuracy = 0.975
데이터 정확도	97.5%

구분 (3-2.3)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 50 - 학습량(epoch) : 200 - 은닉 계층 수 / 폭 : 2 / [12, 6]
결과화면	은닉 계층 2개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 20: loss=0.186, result=0.909,0.000,0.000,0.000 Epoch 40: loss=0.154, result=0.909,0.000,0.000,0.000 Epoch 60: loss=0.142, result=0.909,0.000,0.000,0.000 Epoch 80: loss=0.133, result=0.909,0.000,0.000,0.000 Epoch 100: loss=0.126, result=0.971,0.951,0.716,0.817 Epoch 120: loss=0.120, result=0.971,0.941,0.732,0.823 Epoch 140: loss=0.116, result=0.972,0.945,0.735,0.827 Epoch 160: loss=0.112, result=0.973,0.943,0.753,0.837 Epoch 180: loss=0.109, result=0.973,0.939,0.756,0.838 Epoch 200: loss=0.106, result=0.974,0.950,0.750,0.838 Final Test: final result = 0.974,0.950,0.750,0.838
데이터 정확도	97.4%

구분 (3-2.4)	내용
조건	- 학습률(learning rate) : 0.09 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 200 - 은닉 계층 수 / 폭 : 2 / [12, 6]
결과화면	은닉 계층 2개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 20: loss=0.694, result=0.503,0.503,1.000,0.669 Epoch 40: loss=0.694, result=0.503,0.503,1.000,0.669 Epoch 60: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 80: loss=0.694, result=0.503,0.503,1.000,0.669 Epoch 100: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 120: loss=0.694, result=0.503,0.503,1.000,0.669 Epoch 140: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 160: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 180: loss=0.694, result=0.503,0.503,1.000,0.669 Epoch 200: loss=0.694, result=0.497,0.000,0.000,0.000 Final Test: final result = 0.497,0.000,0.000,0.000
데이터 정확도	49.7%

구분 (3-2.5)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 200 - 은닉 계층 수 / 폭 : 3 / [12, 6, 4]
결과화면	은닉 계층 3개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 20: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 40: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 60: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 80: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 100: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 120: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 140: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 160: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 180: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 200: loss=0.267, result=0.926,0.944,0.908,0.925 Final Test: final result = 0.926,0.944,0.908,0.925
데이터 정확도	92.6%

구분 (3-2.6)	내용
조건	- 학습률(learning rate) : 0.09 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 100 - 은닉 계층 수 / 폭 : 3 / [12, 6, 4]
결과화면	은닉 계층 3개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 10: loss=0.694, result=0.503,0.503,1.000,0.670 Epoch 20: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 30: loss=0.694, result=0.503,0.503,1.000,0.670 Epoch 40: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 50: loss=0.693, result=0.497,0.000,0.000,0.000 Epoch 60: loss=0.694, result=0.503,0.503,1.000,0.670 Epoch 70: loss=0.694, result=0.503,0.503,1.000,0.670 Epoch 80: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 90: loss=0.694, result=0.497,0.000,0.000,0.000 Epoch 100: loss=0.694, result=0.503,0.503,1.000,0.670 Final Test: final result = 0.503,0.503,1.000,0.670
데이터 정확도	50.3%

[천체 펄서 판정 문제 - 실험 결과]
모두 6번의 실험을 진행 하였으며, 1번째 실험의 경우 0개의 은닉 계층을 가지도록 설정 하였습니다. 따라서 이전의 과제와 동일한 결과를 볼 수 있었으며, 이 후의 실험에서는 은닉 계층과 하이퍼 파라미터의 변화를 주었습니다. 균형이 잡히지 않은 데이터셋이다 보니 변화를 주어도 정확도의 범위가 크게 변하지 않는 모습을 확인할 수 있었습니다.

(3-2.3,4,5,6) 실험의 경우 균형 잡힌 데이터 셋을 이용하여 실험을 진행하였고, 은닉 계층과 하이퍼 파라미터에 변화를 주었습니다. 균형 잡힌 데이터 셋을 이용하니 이전의 실험보다 정확한 정확도를 볼 수 있었고, 다음과 같은 특이점을 발견할 수 있었습니다.

은닉 계층 수 / 폭 : 2 / [12 , 6] 학습률 : 0.09
은닉 계층 수 / 폭 : 3 / [12 , 6, 4] 학습률 : 0.09
의 조건 속에서 정확도는 이전의 실험보다 현저히 떨어지는 모습(49.7%)을 확인할 수 있었습니다.

우리는 이를 통해 이전의 [전복의 고리 수 추정]에서와 동일하게 적절한 은닉 계층의 수와 폭 , 적절한 하이퍼 파라미터 값의 조정이 신경망의 성능에 끼치는 영향이 얼마나 중요한지 확인 할 수 있었습니다.

[철판 재료의 불량 판별 문제] 3-3

구분 (3-3.1)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 0 / 0
결과화면	은닉 계층 0개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=15.984, accuracy=0.306/0.320 Epoch 2: loss=15.509, accuracy=0.326/0.197 Epoch 3: loss=15.984, accuracy=0.306/0.348 Epoch 4: loss=15.004, accuracy=0.348/0.197 Epoch 5: loss=15.286, accuracy=0.336/0.202 Epoch 6: loss=15.390, accuracy=0.332/0.440 Epoch 7: loss=15.509, accuracy=0.326/0.442 Epoch 8: loss=15.628, accuracy=0.321/0.455 Epoch 9: loss=15.360, accuracy=0.333/0.322 Epoch 10: loss=15.316, accuracy=0.335/0.455 Final Test: final accuracy = 0.455
데이터 정확도	45.5%

구분 (3-3.2)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 1 / 10
결과화면	은닉 계층 하나를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=2.277, accuracy=0.348/0.325 Epoch 2: loss=1.928, accuracy=0.352/0.325 Epoch 3: loss=1.916, accuracy=0.352/0.325 Epoch 4: loss=1.905, accuracy=0.352/0.325 Epoch 5: loss=1.895, accuracy=0.352/0.325 Epoch 6: loss=1.885, accuracy=0.352/0.325 Epoch 7: loss=1.875, accuracy=0.352/0.325 Epoch 8: loss=1.866, accuracy=0.352/0.325 Epoch 9: loss=1.858, accuracy=0.352/0.325 Epoch 10: loss=1.849, accuracy=0.352/0.325 Final Test: final accuracy = 0.325
데이터 정확도	32.5%

구분 (3-3.3)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 50 - 은닉 계층 수 / 폭 : 3 / [12, 6, 4]
결과화면	은닉 계층 3개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 5: loss=1.896, accuracy=0.351/0.330 Epoch 10: loss=1.850, accuracy=0.351/0.330 Epoch 15: loss=1.814, accuracy=0.351/0.330 Epoch 20: loss=1.786, accuracy=0.351/0.330 Epoch 25: loss=1.764, accuracy=0.351/0.330 Epoch 30: loss=1.746, accuracy=0.351/0.330 Epoch 35: loss=1.731, accuracy=0.357/0.330 Epoch 40: loss=1.722, accuracy=0.351/0.330 Epoch 45: loss=1.714, accuracy=0.351/0.330 Epoch 50: loss=1.707, accuracy=0.351/0.330 Final Test: final accuracy = 0.330
데이터 정확도	33.0%

구분 (3-3.4)	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 50 - 은닉 계층 수 / 폭 : 4 / [12, 6, 4, 2]
결과화면	은닉 계층 4개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 5: loss=1.895, accuracy=0.348/0.340 Epoch 10: loss=1.849, accuracy=0.348/0.340 Epoch 15: loss=1.814, accuracy=0.348/0.340 Epoch 20: loss=1.786, accuracy=0.348/0.340 Epoch 25: loss=1.764, accuracy=0.348/0.340 Epoch 30: loss=1.747, accuracy=0.348/0.340 Epoch 35: loss=1.733, accuracy=0.348/0.340 Epoch 40: loss=1.722, accuracy=0.348/0.340 Epoch 45: loss=1.714, accuracy=0.348/0.340 Epoch 50: loss=1.707, accuracy=0.348/0.340 Final Test: final accuracy = 0.340
데이터 정확도	34.0%

구분 (3-3.5)	내용
조건	- 학습률(learning rate) : 0.0001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 50 - 은닉 계층 수 / 폭 : 3 / [12, 6, 4]
결과화면	은닉 계층 3개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 5: loss=1.777, accuracy=0.335/0.304 Epoch 10: loss=1.649, accuracy=0.253/0.243 Epoch 15: loss=1.584, accuracy=0.432/0.460 Epoch 20: loss=1.551, accuracy=0.441/0.437 Epoch 25: loss=1.541, accuracy=0.446/0.427 Epoch 30: loss=1.558, accuracy=0.435/0.440 Epoch 35: loss=1.533, accuracy=0.442/0.465 Epoch 40: loss=1.575, accuracy=0.421/0.450 Epoch 45: loss=1.560, accuracy=0.459/0.389 Epoch 50: loss=1.561, accuracy=0.430/0.471 Final Test: final accuracy = 0.471
데이터 정확도	47.1%

구분 (3-3.6)	내용
조건	- 학습률(learning rate) : 0.0001 - 매개변수 인수값(mb_size) : 50 - 학습량(epoch) : 100 - 은닉 계층 수 / 폭 : 3 / [12, 6, 4]
결과화면	은닉 계층 3개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 10: loss=1.825, accuracy=0.340/0.373 Epoch 20: loss=1.767, accuracy=0.297/0.228 Epoch 30: loss=1.637, accuracy=0.222/0.205 Epoch 40: loss=1.560, accuracy=0.429/0.463 Epoch 50: loss=1.555, accuracy=0.419/0.455 Epoch 60: loss=1.542, accuracy=0.439/0.486 Epoch 70: loss=1.526, accuracy=0.415/0.440 Epoch 80: loss=1.517, accuracy=0.425/0.460 Epoch 90: loss=1.521, accuracy=0.427/0.442 Epoch 100: loss=1.521, accuracy=0.435/0.481 Final Test: final accuracy = 0.481
데이터 정확도	48.1%

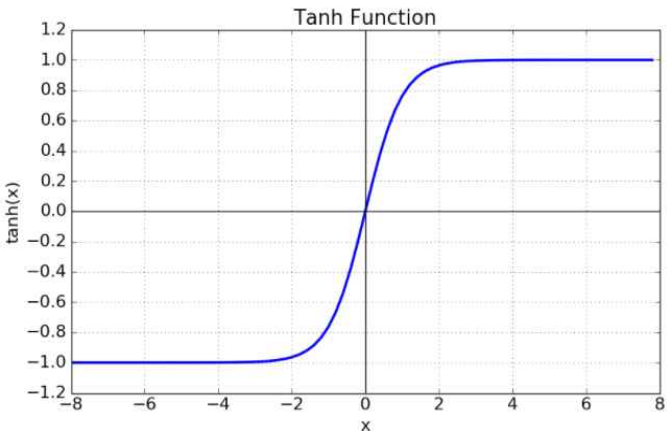
[철판 재료의 불량 판별 문제 - 실험 결과]
모두 6번의 실험을 진행 하였으며, 1번째 실험의 경우 0개의 은닉 계층을 가지도록 설정 하였습니다. 따라서 이전의 과제와 동일한 결과를 볼 수 있었으며, 이 후의 실험에서는 은닉 계층과 하이퍼 파라미터의 변화를 주었습니다.

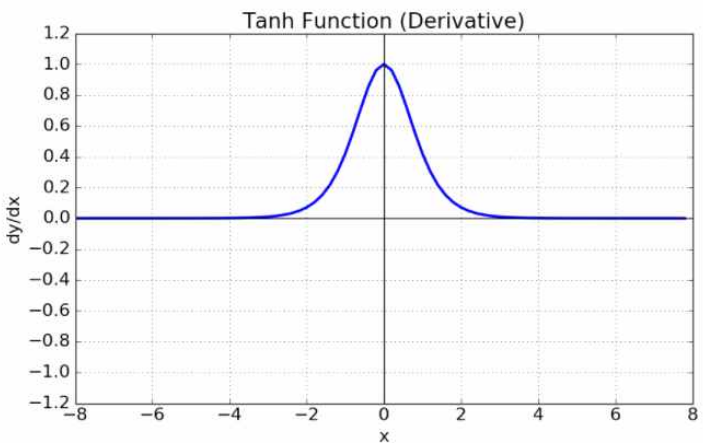
특이점은 학습률을 0.0001로 변경하였을 때, 45%이상의 정확도를 볼 수 있었으며, 3-3.6의 조건에서는 48.1%의 최대 정확도를 확인할 수 있었습니다. 하지만 여전히 하이퍼 파라미터의 변화 또한 성능 변화에 큰 영향을 끼치는 모습을 실험을 통해 확인하였습니다.

[최종 결론]
다층 퍼셉트론 도입으로 약간의 성능 향상 효과를 확인할 수 있었습니다. 무조건 적인 은닉 계층의 수와 폭 추가는 성능 향상으로 이어지는 것이 아니라는 것을 실험을 통해 확인 하였으며, 하이퍼 파라미터 , 은닉 계층의 수와 폭 이 모든 것이 적절한 조화를 이루어야 성능향상이 큰 폭으로 이루어진다는 사실을 확인하였습니다.

이에 다층 퍼셉트론의 도입은 선형 연산에 가까운 처리만으로 좋은 결과를 얻을 수 없는 문제에 유용한 것을 알 수 있었으며, 이론에서 보았던 것과 같이 이러한 문제에서는 많은 양의 학습과 함께 입력의 분포 양상을 보여줄 더 많은 양의 데이터가 필요하다는 것을 알 수 있었습니다.

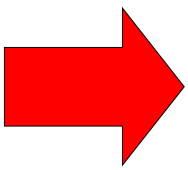
[tanh 함수 - Hyperbolic tangent function]
tanh 함수는 sigmoid의 대체할 수 있는 활성화 함수입니다. sigmoid 함수와 매우 유사하며, 그래프는 다음과 같습니다.





기존의 ReLU 함수의 코드를 tanh함수의 코드로 변형시켜 실험을 3번 진행하였습니다.

```
def relu(x):
    return np.maximum(x,0)
```



```
def tanh(x):
    return np.tanh(x)
```

구분 (3.1) - 전복의 고리 수 추정	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 10 - 은닉 계층 수 / 폭 : 4 / [2, 2, 2, 2]
결과화면	은닉 계층 4개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 1: loss=65.259, accuracy=0.299/0.526 Epoch 2: loss=24.925, accuracy=0.644/0.723 Epoch 3: loss=14.322, accuracy=0.742/0.757 Epoch 4: loss=11.548, accuracy=0.753/0.752 Epoch 5: loss=10.813, accuracy=0.744/0.743 Epoch 6: loss=10.624, accuracy=0.738/0.738 Epoch 7: loss=10.572, accuracy=0.734/0.735 Epoch 8: loss=10.558, accuracy=0.733/0.734 Epoch 9: loss=10.555, accuracy=0.731/0.733 Epoch 10: loss=10.554, accuracy=0.731/0.733 Final Test: final accuracy = 0.733
데이터 정확도	73.3%

[전복의 고리 수 판정 - 위의 조건에서 정확히는 tanh 함수가 ReLU 함수에 비해 더 나은 성능을 보이고 있으나,
두 함수의 성능이 비슷함을 확인할 수 있습니다.]

구분 (3-2.5) - 천체 펄서 판정 문제	내용
조건	- 학습률(learning rate) : 0.001 - 매개변수 인수값(mb_size) : 10 - 학습량(epoch) : 200 - 은닉 계층 수 / 폭 : 3 / [12, 6, 4]
결과화면	은닉 계층 3개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 20: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 40: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 60: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 80: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 100: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 120: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 140: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 160: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 180: loss=0.307, result=0.909,0.000,0.000,0.000 Epoch 200: loss=0.307, result=0.909,0.000,0.000,0.000 Final Test: final result = 0.909,0.000,0.000,0.000
데이터 정확도	90.9%

[천체 펄서 판정 문제 - 위의 조건에서는 tanh 함수에 비해서 ReLU 함수가 더 나은 성능을 보이고 있다.]

구분 (3-3.6) - 철판 재료의 불량 판별 문제	내용
조건	- 학습률(learning rate) : 0.0001 - 매개변수 인수값(mb_size) : 50 - 학습량(epoch) : 100 - 은닉 계층 수 / 폭 : 3 / [12, 6, 4]
결과화면	은닉 계층 3개를 갖는 다층 퍼셉트론이 작동되었습니다. Epoch 10: loss=1.944, accuracy=0.341/0.368 Epoch 20: loss=1.941, accuracy=0.341/0.368 Epoch 30: loss=1.939, accuracy=0.341/0.368 Epoch 40: loss=1.937, accuracy=0.341/0.368 Epoch 50: loss=1.934, accuracy=0.341/0.368 Epoch 60: loss=1.932, accuracy=0.341/0.368 Epoch 70: loss=1.930, accuracy=0.341/0.368 Epoch 80: loss=1.928, accuracy=0.341/0.368 Epoch 90: loss=1.925, accuracy=0.341/0.368 Epoch 100: loss=1.923, accuracy=0.341/0.368 Final Test: final accuracy = 0.368
데이터 정확도	36.8%

[철판 재료의 불량 판별 문제 - 위의 조건에서는 tanh 함수에 비해서 ReLU 함수가 더 나은 성능을 보이고 있다.]

결론 : 대체적으로 ReLU 함수의 정확도가 조금 더 높거나 비슷한 것을 알 수 있다. 그러나 이것이 항상 성립하는 것은 아니기 때문에 주어진 문제에 대해서 test를 해보아야 하며, 신경망의 구조, 주어진 문제에 따라 적절한 활성화 함수를 사용하는 것이 바람직하다.