04 예측함수 정의하기

예측함수 = 레이어 함수의 조합

레이어 함수=빌딩블록, 레이어, 모듈

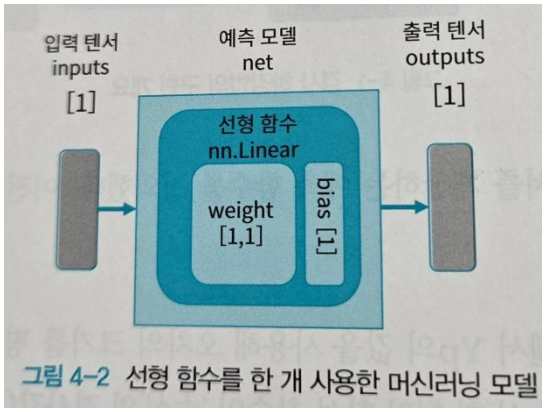


그림 4-2 선형 함수를 한 개 사용한 머신러닝 모델

선형함수 : 일차 함수를 일반화한 것

=> $Y_p = W * X + B$

=> 두 개의 파라미터(W, B)

=> 파이토치에서 nn.Linear로 사용

일반적으로는 아래와 같이 여러 층을 조합해 예측함수 구성

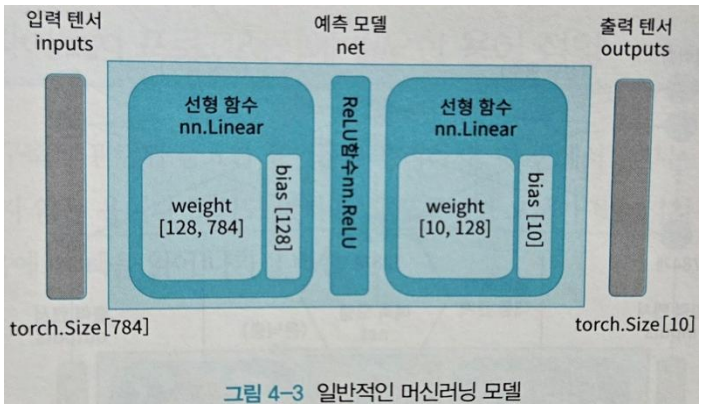


그림 4-3 일반적인 머신러닝 모델

'선형함수' 두 번 사용

'ReLU함수' 한 번 사용

용어 정리

파라미터: 레이어 함수의 내부에서 가지고 있는 입력 텐서 이외의 데이터

- 레이어 함수의 파라미터를 조정하는 것 = 학습

입력 텐서: 신경망 개념에서 입력층에 해당

출력 텐서: 신경망 개념에서 출력층에 해당

머신러닝 모델: 여러 개의 레이어 함수를 조합해 입력 텐서에 대해 바람직한 출력텐서를 출력해주는 합성함수.

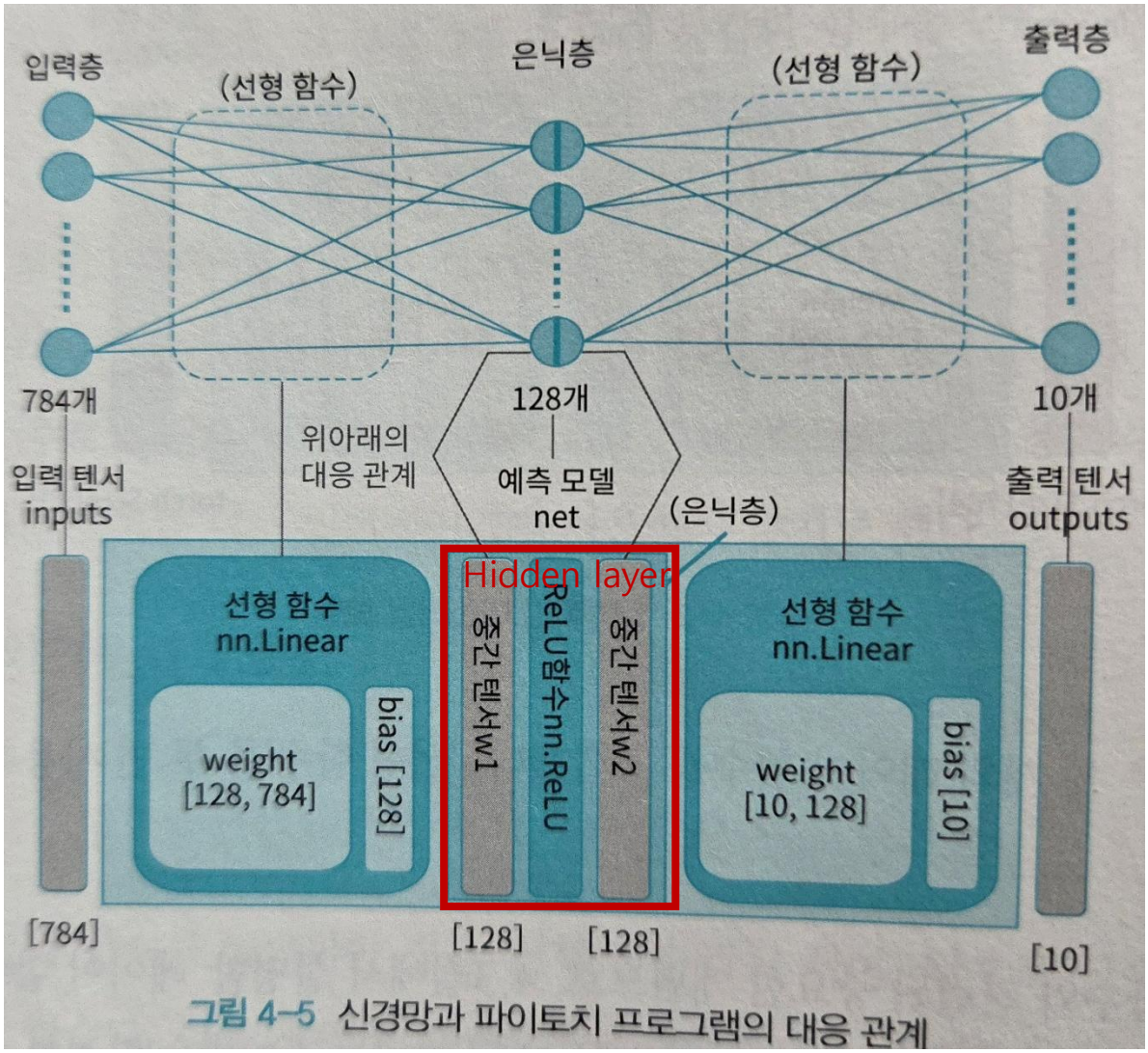


그림 4-5 신경망과 파이토치 프로그램의 대응 관계

파이토치 온라인 문서에는 활성화 함수를 레이어 함수로 정의하지 않지만 이 책에서는 활성화 함수도 레이어 함수와 같은 취급.

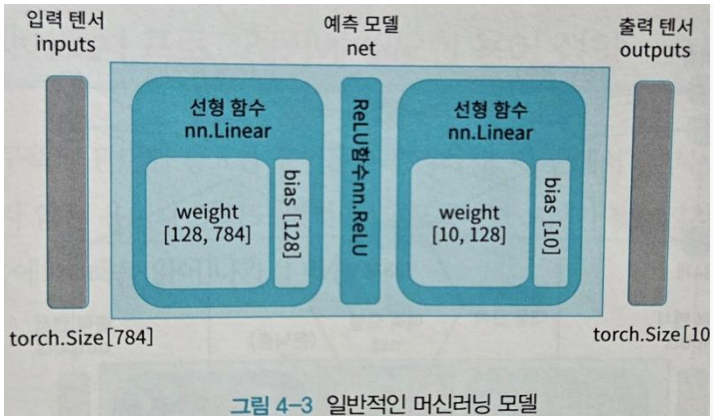


그림 4-3 일반적인 머신러닝 모델

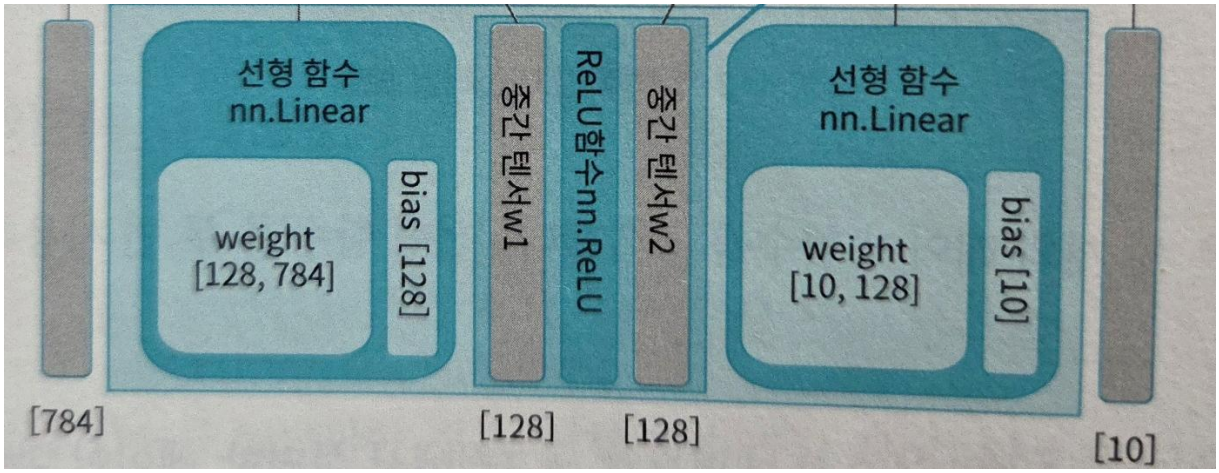
128이 은닉층의 노드수임!

```
#코드 4-1) 레이어 함수 정의

#첫번째 선형 함수
#입력수:784, 출력수:128
l1 = nn.Linear(784, 128)

#두번째 선형 함수
#입력수:128, 출력수:10
l2 = nn.Linear(128, 10)

#활성화 함수
relu = nn.ReLU(inplace=True)
```




```
#코드 4-2) 입력 텐서로부터 출력 텐서를 계산

# 더미 입력 데이터 작성
inputs = torch.randn(100, 784)

# 중간 텐서 1 계산
m1 = l1(inputs)

# 중간 텐서 2 계산
m2 = relu(m1)

# 출력 텐서 계산
outputs = l2(m2)

# 입력 텐서와 출력 텐서 shape 확인
print('입력 텐서', inputs.shape)
print('출력 텐서', outputs.shape)
```

```
입력 텐서 torch.Size([100, 784])
출력 텐서 torch.Size([100, 10])
```

```
#코드 4-3) nn.Sequential을 사용해 전체를 합성 함수로 정의

net2 = nn.Sequential(
    l1,
    relu,
    l2
)

outputs2 = net2(inputs)

# 입력 텐서와 출력 텐서의 shape 확인
print('입력 텐서', inputs.shape)
print('출력 텐서', outputs2.shape)
```

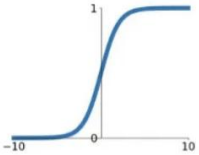
```
입력 텐서 torch.Size([100, 784])
출력 텐서 torch.Size([100, 10])
```

nn.Sequential을 통해 중간 텐서 생략

-활성화 함수의 목적

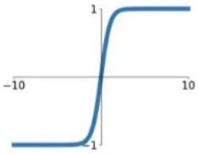
Sigmoid

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



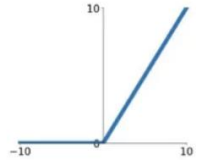
tanh

$$\tanh(x)$$



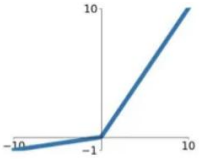
ReLU

$$\max(0, x)$$



Leaky ReLU

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

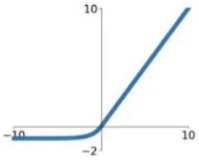


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

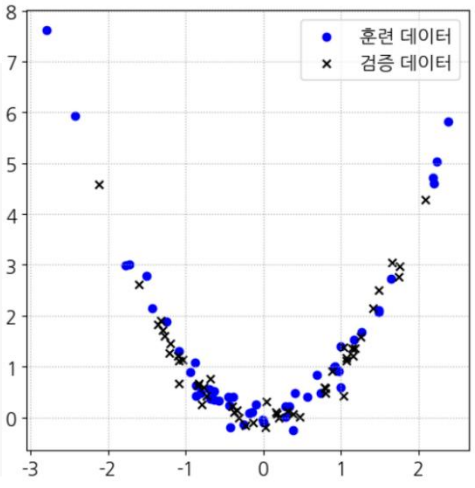
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



```python
#코드 4-4) 훈련 데이터, 검증 데이터 계산
np.random.seed(123)
x = np.random.randn(100,1)

# y는 x^2에 난수를 1/10만큼 더한 값
y = x**2 + np.random.randn(100,1) * 0.2

# 데이터를 50건씩 훈련용과 검증용으로 나눔
x_train = x[:50,:]
x_test = x[50:,:]
y_train = y[:50,:]
y_test = y[50:,:]
```




```python
#코드 4-5) 산포도 출력
plt.scatter(x_train, y_train, c='b', label='training set')
plt.scatter(x_test, y_test, c='k', marker='x', label='test set')
plt.legend()
plt.show()
```

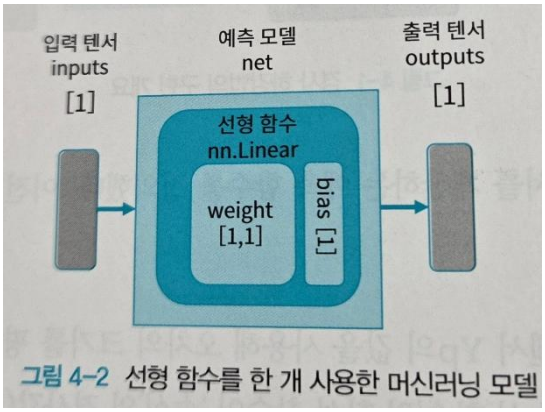
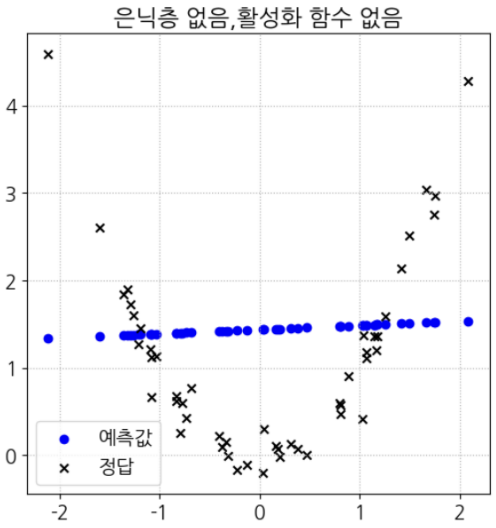


그림 4-2 선형 함수를 한 개 사용한 머신러닝 모델



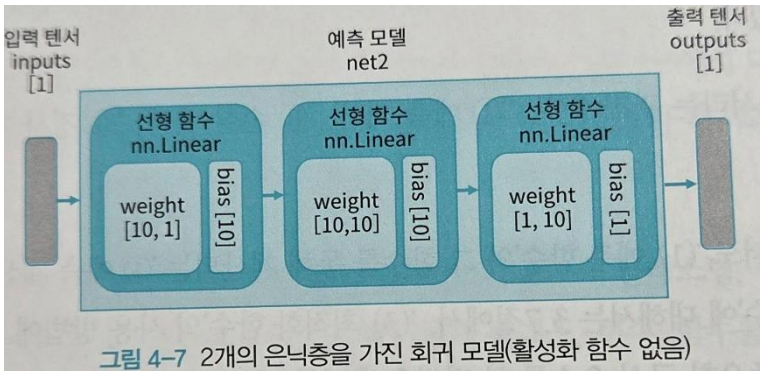
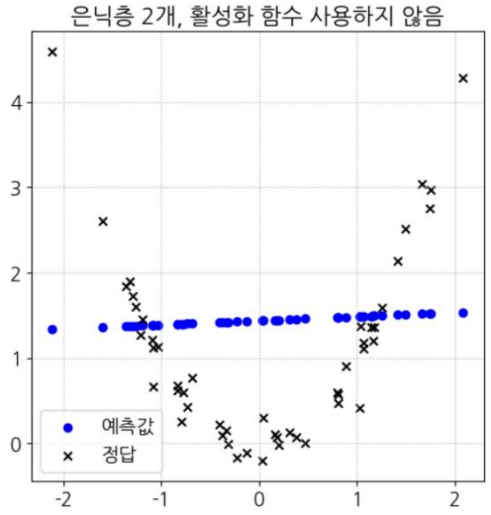


그림 4-7 2개의 은닉층을 가진 회귀 모델(활성화 함수 없음)



단순히 선형 함수를 합성해 놓기만 한 함수는 결국 한 개 층의 선형함수와 같음

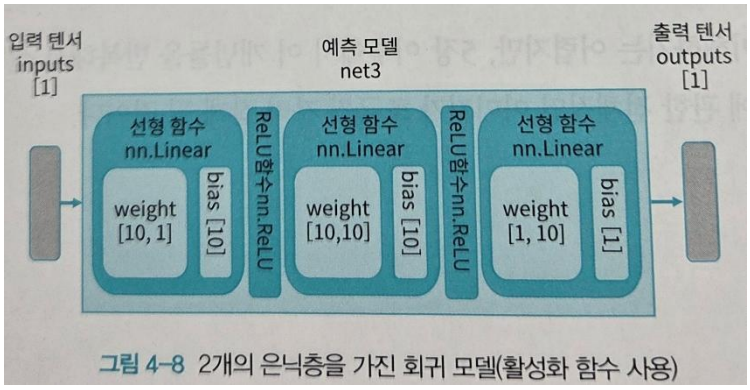
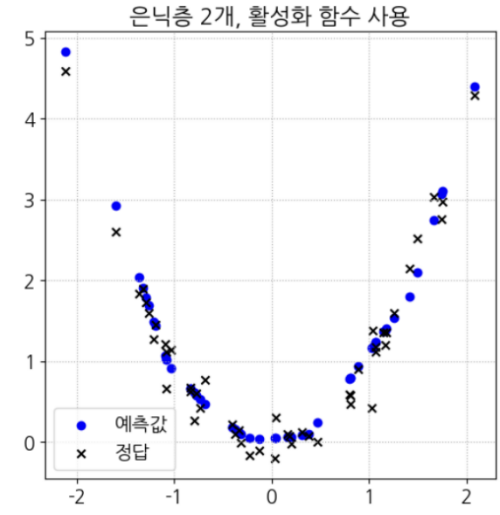


그림 4-8 2개의 은닉층을 가진 회귀 모델(활성화 함수 사용)



⇨ '비선형 함수'로 불리는 활성화 함수를 선형 함수 사이에 넣어야 비로소 깊은 층을 가진 딥러닝 모델이 의미를 가짐