2,3,4와 마찬가지로 입출력차원이 각각 2인 선형층을 쌓고 마지막 출력까지 총 5개의 히든레이어로 이루어진 모델을 설명하겠다. 기존과 동일하게 최종적으로 한 개의 출력을 내놓고자하는 것이 목표고, 그러기 위해 모델 객체 생성 > 모델 학습 > 모델을 사용하여 적절한 파라미터값(입력데이터에 관한 예측치)을 출력하는 과정을 거친다.

```
import torch


# 주어진 파라미터
w0 = torch.tensor([[16.9412, -8.1639], [10.8629, -6.3703]])
b0 = torch.tensor([ 2.3297, -3.8986])
w2 = torch.tensor([[-4.8914, 3.4389], [3.8840, -4.5790]])
b2 = torch.tensor([ 0.8771, -0.5586])
w4 = torch.tensor([[ 5.1687, -4.6577],[-3.9116, 3.9123]])
b4 = torch.tensor([ 0.0831, -0.1589])
w6 = torch.tensor([[-5.0883, 4.3056], [-4.2708, 3.9464]])
b6 =  torch.tensor([-0.3551, -0.5728])
w8 = torch.tensor([[-5.3847, -4.3799], [4.3775, 4.7031]])
b8 = torch.tensor([[4.3330, -3.9977]])
w10 = torch.tensor([[10.7038, -10.0231]])
b10 = torch.tensor([[-0.1516]])

# 입력 데이터
X = torch.tensor([[0, 0], [0, 1], [1, 0], [1, 1]],
dtype=torch.float32)

sigmoid = nn.Sigmoid()

# 첫 번째 은닉층의 출력 계산
layer1_output = sigmoid(X @ w0.T + b0)

# 두 번째 은닉층의 출력 계산
layer2_output = sigmoid(layer1_output @ w2.T + b2)

# 세 번째 은닉층의 출력 계산
layer3_output = sigmoid(layer2_output @ w4.T + b4)

# 네 번째
layer4_output = sigmoid(layer3_output @ w6.T + b6)

#다섯번째
layer5_output = sigmoid(layer4_output @ w8.T + b8)
```

```
# 최종 출력 계산
final_output = sigmoid(layer5_output @ w10.T + b10)
print(final_output)
```

최종출력을 계산하기위해 앞서서 정의한 파라미터와 입력데이터에 기반한 순전파를 진행한다. 주어진 파라미터는 각 층의 가중치와 편향을 뜻하고, 입력데이터는 XOR연산을 위한 4가지의 입력조합이다.

각 은닉층과 출력층을 통과할 때마다 시그모이드 함수를 적용하며 각 층의 출력을 계산하고 최종출력을 뽑아내려고 한다.

다시 처음부터 정리하자면

1. XOR 게이트는 두 입력이 서로 다를 때에만 출력이 1이 되는 논리 연산이다. 이를 신경망으로 모델링하기 위해 XOR 클래스가 정의되어 있다. 이 클래스는 여러 개의 선형 층과 시그모이드 활성화 함수로 이루어져 있다.

2. 그 다음, 입력 공간을 50x50 그리드로 나누고, 각 점에서의 모델의 출력을 계산하여 예측값을 얻는다. 이렇게 얻은 예측값을 3차원 공간에 시각화하여 XOR 게이트의 출력을 보여준다.

출력결과는 다음과 같다