Week 01_hw

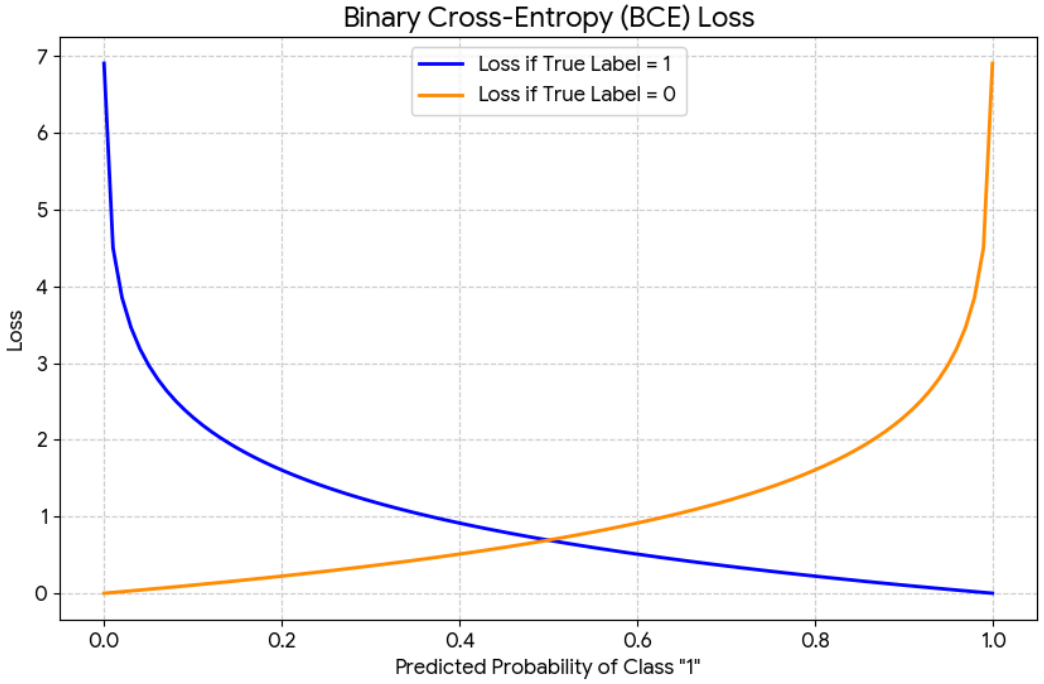
ANN을 이용한 OR Logic Gate Implementation

핵심 코드 분석

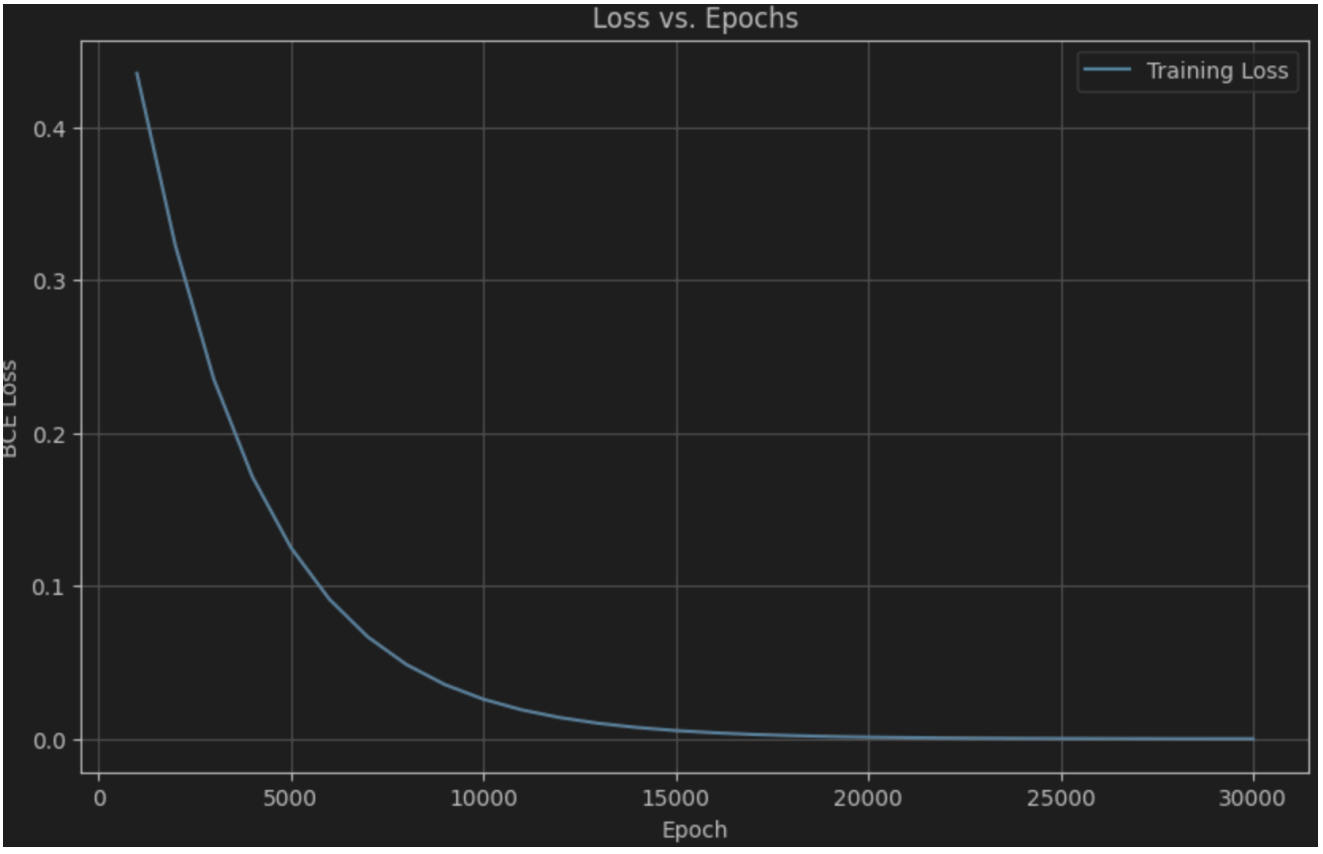
```
class ANN(nn.Module):
    def __init__(self):
        super(ANN, self).__init__()
        self.fc1 = nn.Linear(in_features=2, out_features=1)

    def forward(self, input_data):
        out = self.fc1(input_data)
        out = F.sigmoid(out)
        return out

model = ANN()

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
loss_function = nn.BCELoss()
```

저의 ANN 구현에서는 다음을 사용하여 Network를 구성 하였습니다.

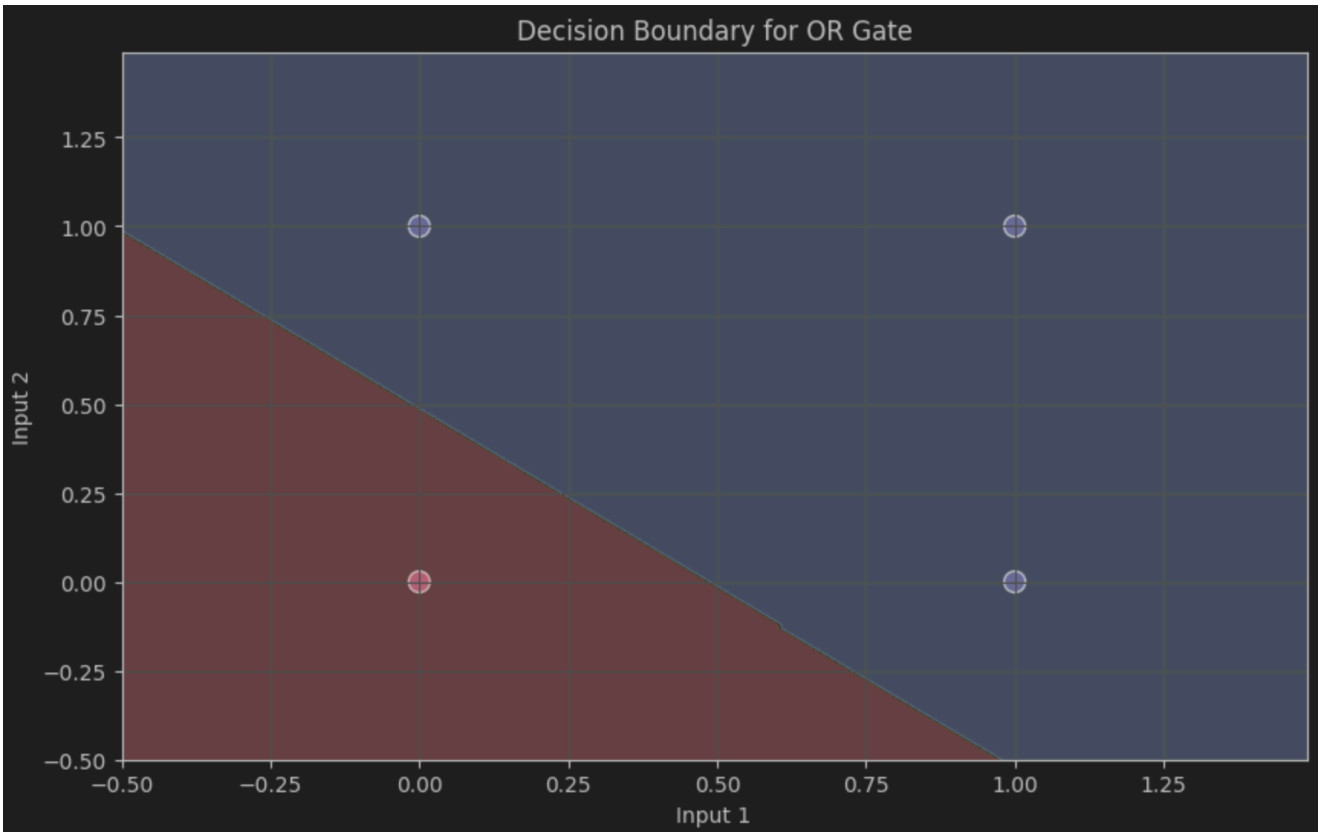
- Activation Function: Sigmoid
- Loss Function: Binary Cross Entropy Loss
- Optimizer: adam
  근거는 다음과 같습니다.
- Binary Label을 가지는 경우로써, BCE 사용시 완전 틀린 경우 아주 강한 신호를 보내서 더 빠르게 학습
- BCE는 0-1까지 input을 받음으로 sigmoid 함수를 사용하였습니다.
- 웬만한 상황에서 좋은 adam을 optimizer로 설정하였습니다.
- Two class problem이여서 multi-class classification이 아닌 binary classification을 사용하였습니다. (computationally efficient)



시각화

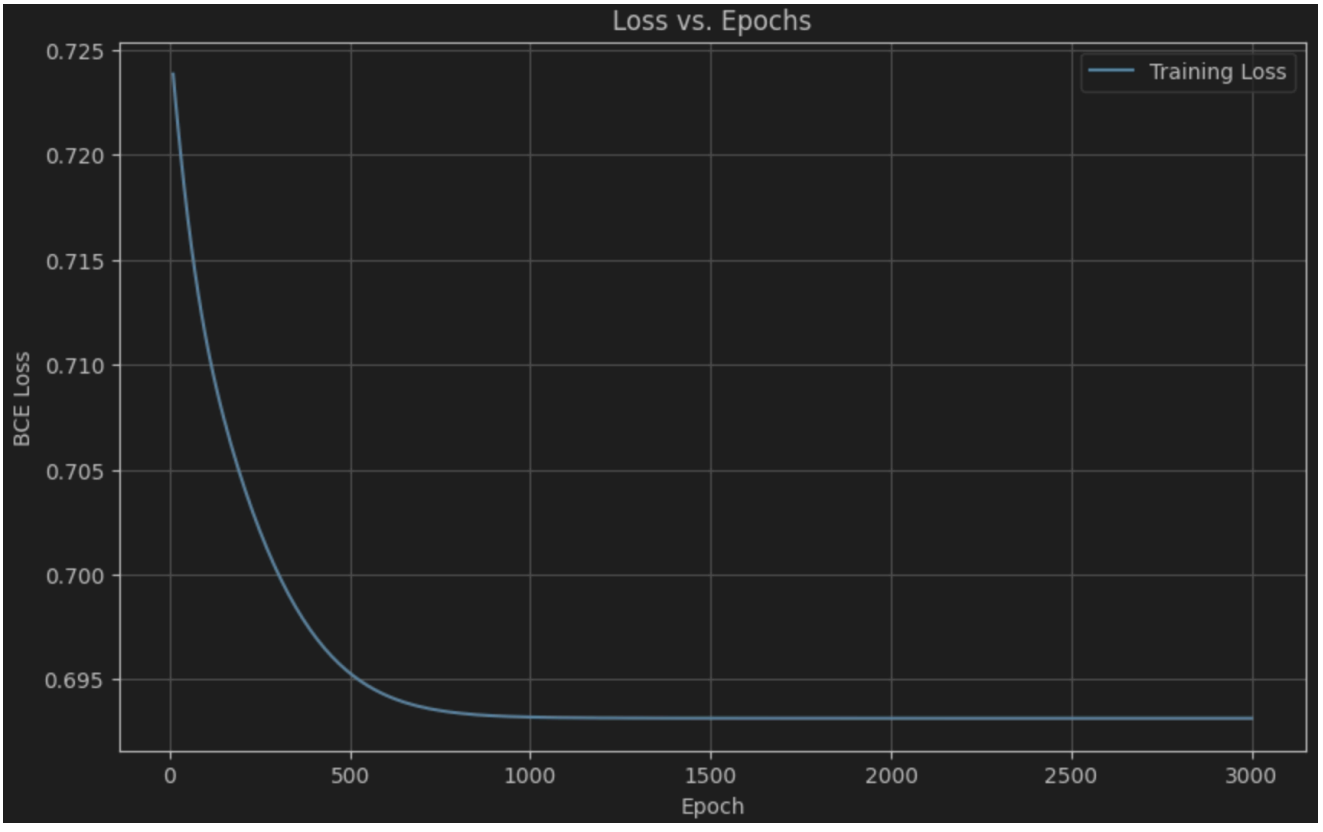


학습이 반복될수록 loss가 0으로 수렴함을 확인할수 있었습니다.

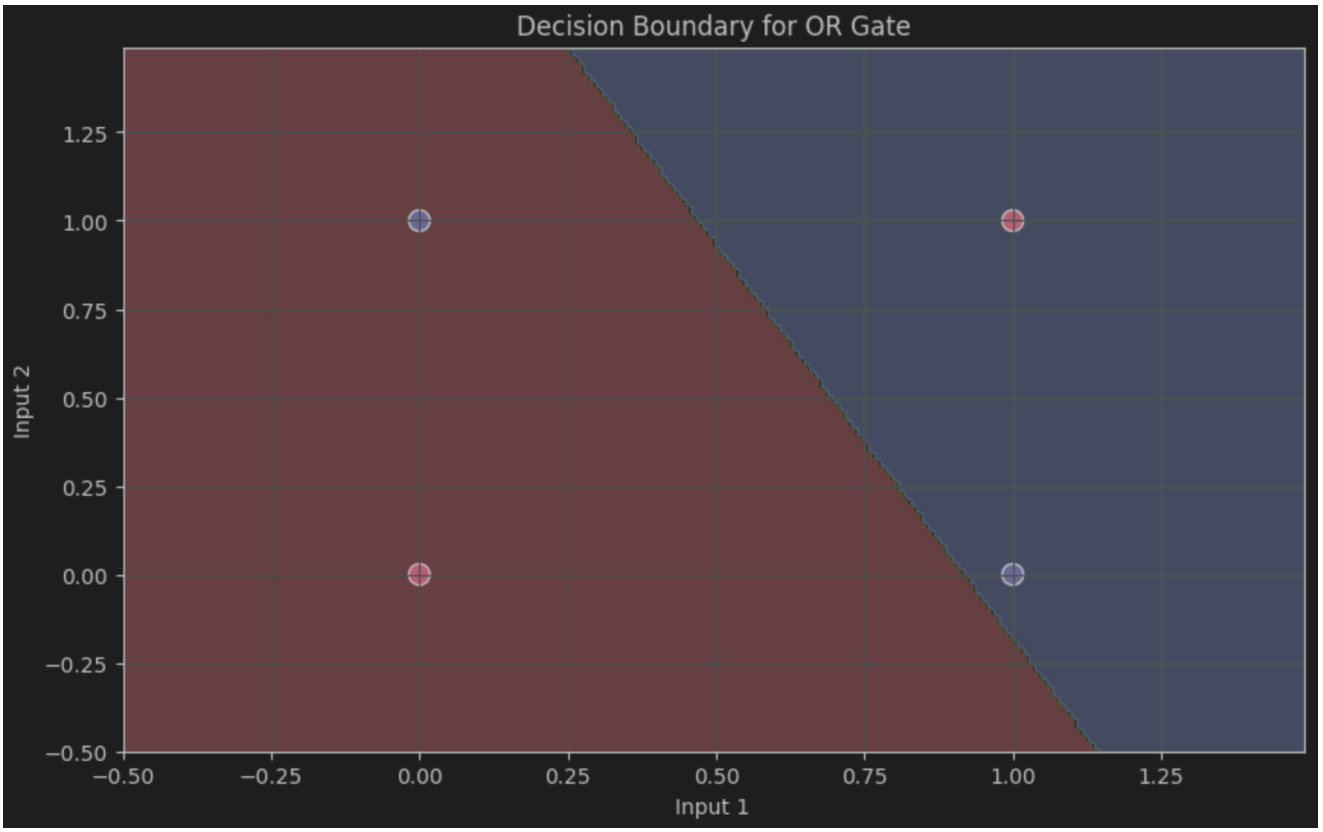


결정 경계가 예쁘게 만들어짐을 확인 했습니다.

ANN이용한 XOR Logic Gate Implementation



아무리 반복해도 loss가 0.6931로 수렴 (0.6931 == BCE(0.5))



아무리 여러번 돌려도 절반만 맞추는 model이 나온다.

분리가 되지 않는 이유

하나의 layer로는 하나의 linear 한 line으로 밖에 decision boundary를 만들지 못해서.
여러 layer을 둠으로써 non-linear한 line을 그릴수 있게 해주자

DNN이용한 XOR Logic Gate Implementation

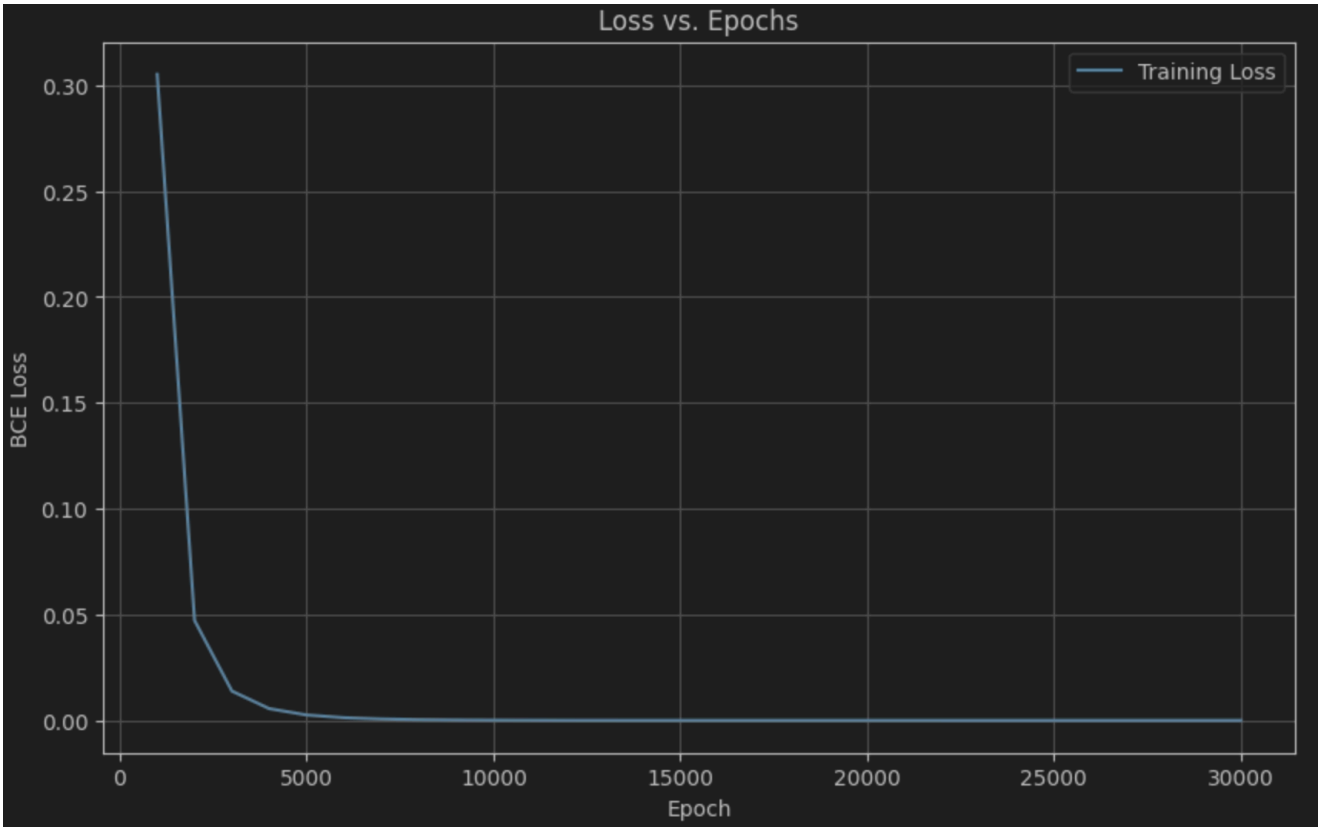
핵심 코드 분석

```
class ANN(nn.Module):
    def __init__(self):
        super(ANN, self).__init__()
        self.fc0 = nn.Linear(in_features=2, out_features=9)
        self.fc1 = nn.Linear(in_features=9, out_features=1)


    def forward(self, input_data):
        out = self.fc0(input_data)
        out = F.leaky_relu(out)
        out = self.fc1(out)
        out = F.sigmoid(out)
        return out

model = ANN()
```

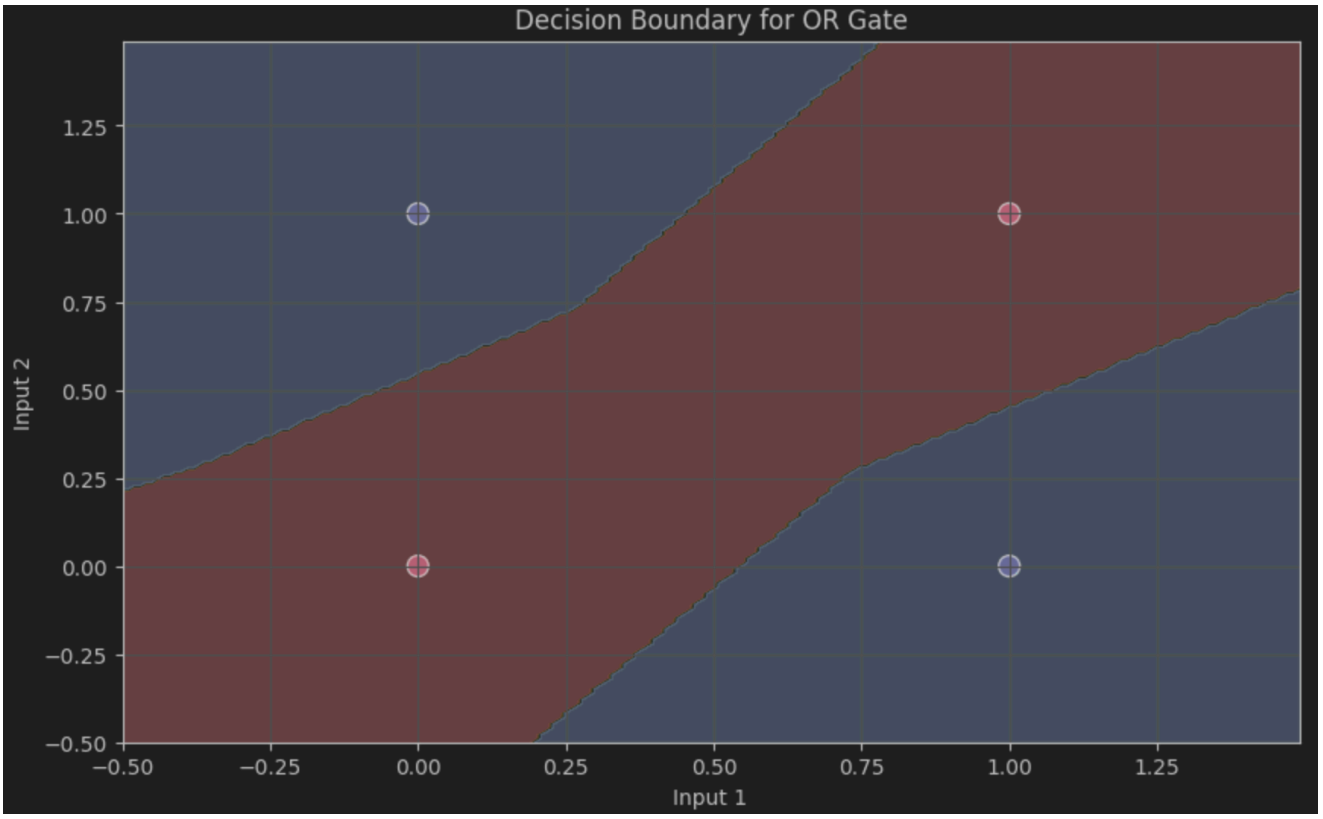
```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
loss_function = nn.BCELoss()
```

- 새로운 layer에 9개의 node를 두었습니다. 2개의 node만으로도 충분한 경우이지만 (b/c 두개의 직선으로 분류가능) local optima에 빠지는 경우가 많아 9개로 설정하여 학습하였습니다.
- 새로운 layer의 activation function으로는 leakyReLU를 사용하였습니다. 이에 따라 sigmoid 보다 더 빠르게 학습할수 있습니다. (b/c stronger derivative for positive inputs)
- 기존 layer의 activation function은 sigmoid로 유지 (b/c BCE의 입력값의 범위는 0-1)

시각화



학습을 함에 따라 0으로 수렴함을 확인할수 있었습니다.



결정 경계가 non linear하게 만들어 짐을 확인 했습니다.

분리가 되는 이유

여러 layer을 둠으로써 non-linear한 결정 경계가 만들어 질수 있었기 때문에