ANN, DNN

날짜	@16/07/2025 → 23/07/2025
Select	과제
Status	Not started
Tags	1차시
자료	25기 BASE 1주차 발표 자료.pptx

OR 게이트 학습 및 시각화 (ANN)

코드 분석

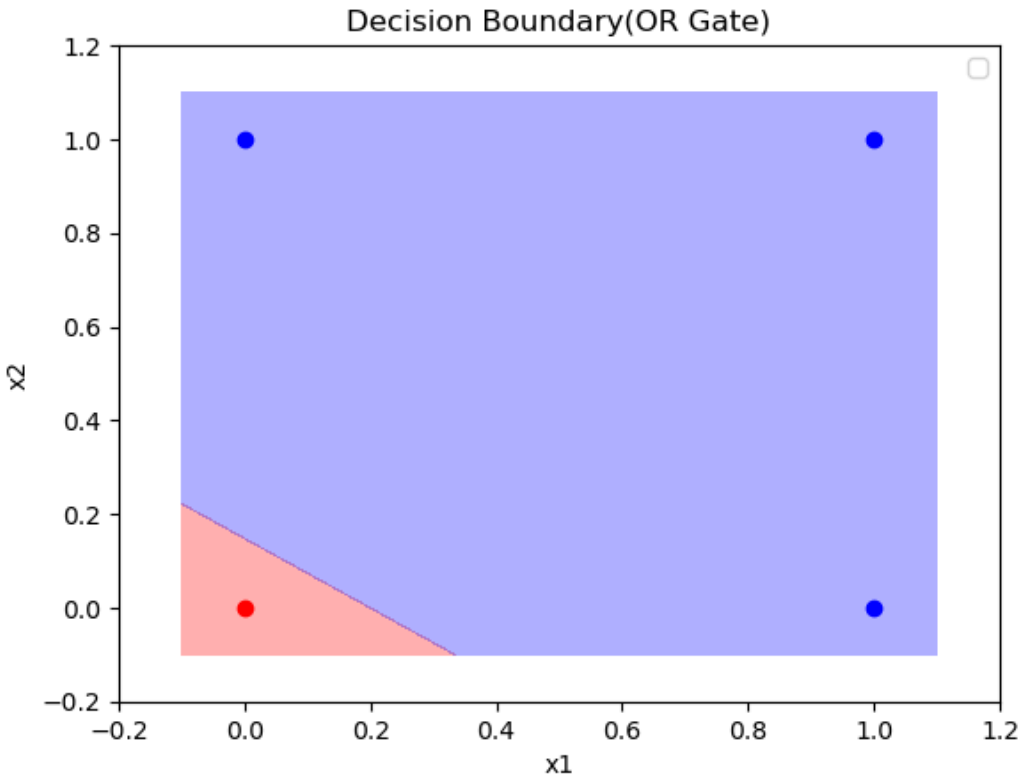
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

# 입력 데이터 생성
data = np.array([[0,0], [0,1], [1,0], [1,1]])
label = np.array([[0], [1], [1], [1]])

# 은닉층이 1개인 모델 생성
model = tf.keras.Sequential([
    tf.keras.layers.Dense(2, activation='relu', input_shape = (2,)),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# 모델 컴파일
model.compile(optimizer='adam',
              loss = 'binary_crossentropy',
              metrics=['accuracy'])
```

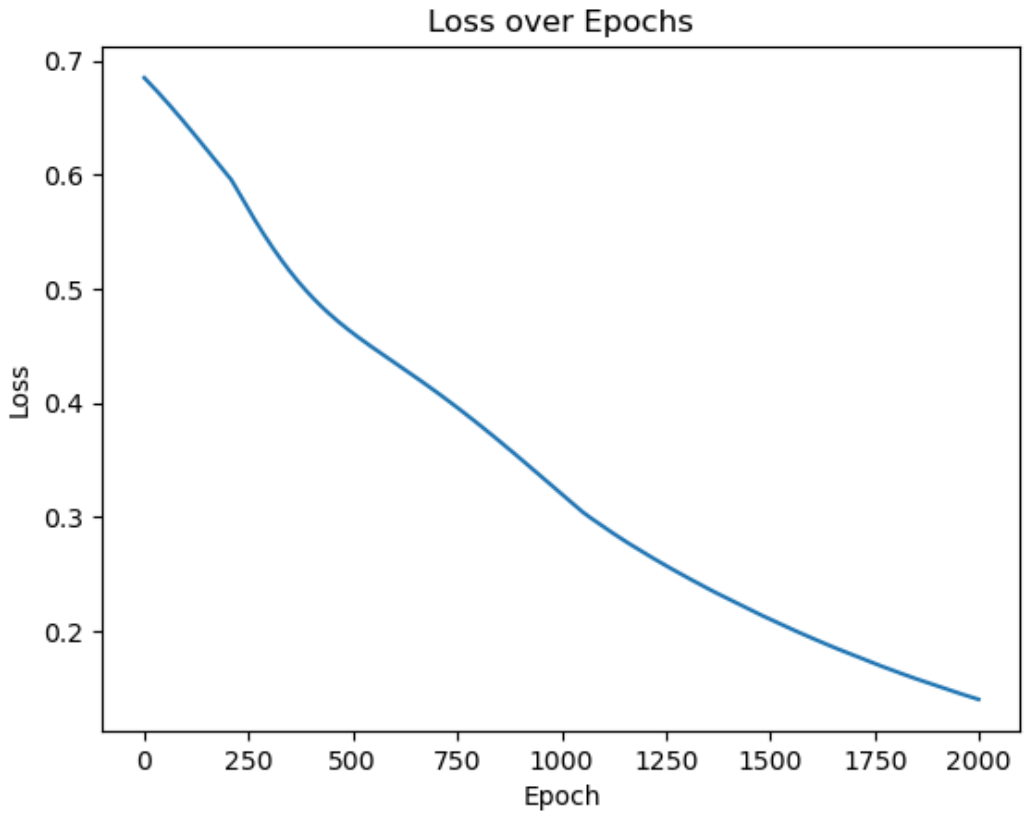
```
history = model.fit(data, label, epochs = 2000)

# 정확도 계산
print("Accuracy: ", model.evaluate(data, label)[1])

# 결정 경계를 위한 meshgrid
xx, yy = np.meshgrid(np.linspace(-0.1, 1.1, 300),
                     np.linspace(-0.1, 1.1, 300))
grid = np.c_[xx.ravel(), yy.ravel()]

# 각 데이터별 확률 출력
Z = model.predict(grid)
for i in range(4):
    print(f"{data[i]} → {grid[i][0]:.4f}")
```

1. 2차원 이진 벡터 형식의 데이터와 라벨 생성
2. 노드가 4개이고 ReLU를 사용하는 하나의 은닉층을 가지는 신경망 구성

   출력층은 노드가 1개로 sigmoid 활성화 함수를 사용해 확률을 출력
3. Adam 옵티마이저와 binary cross entropy loss를 사용해 컴파일
4. 전체 데이터를 사용해 1000번 반복 학습
5. 학습한 모델 정확도 평가

시각화


Decision Boundary(OR Gate)
x1
x2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Loss over Epochs
Loss
Epoch
0.7
0.6
0.5
0.4
0.3
0.2
0
250
500
750
1000
1250
1500
1750
2000

XOR 게이트 학습 시도 및 한계 시각화 (ANN)

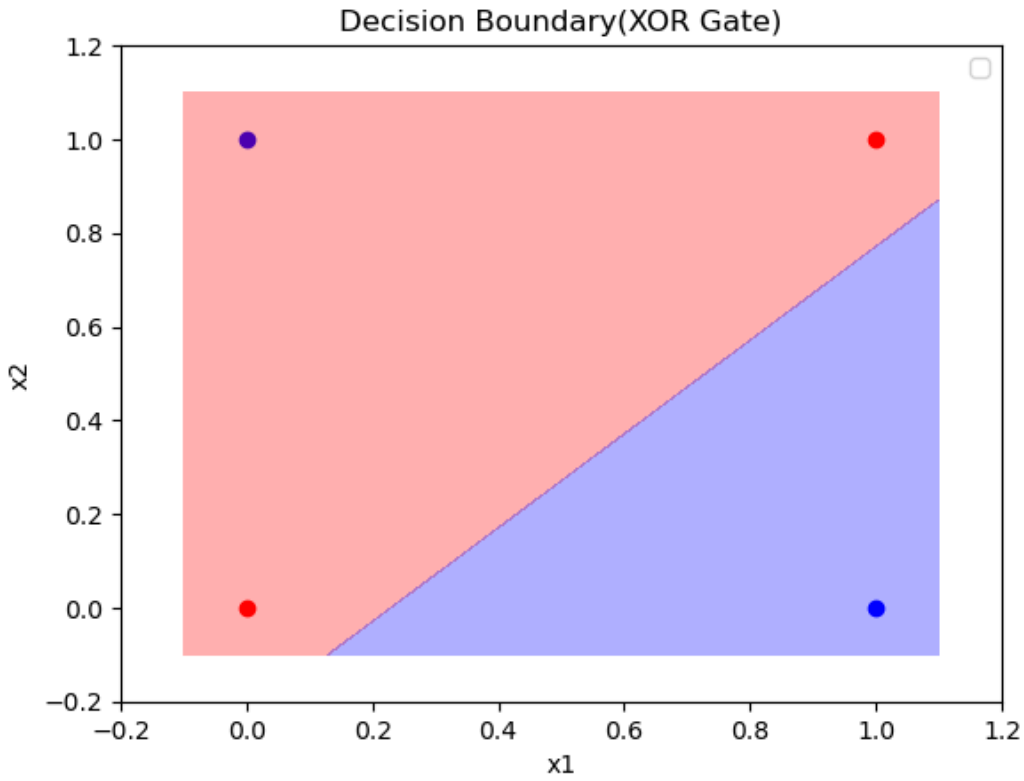
코드

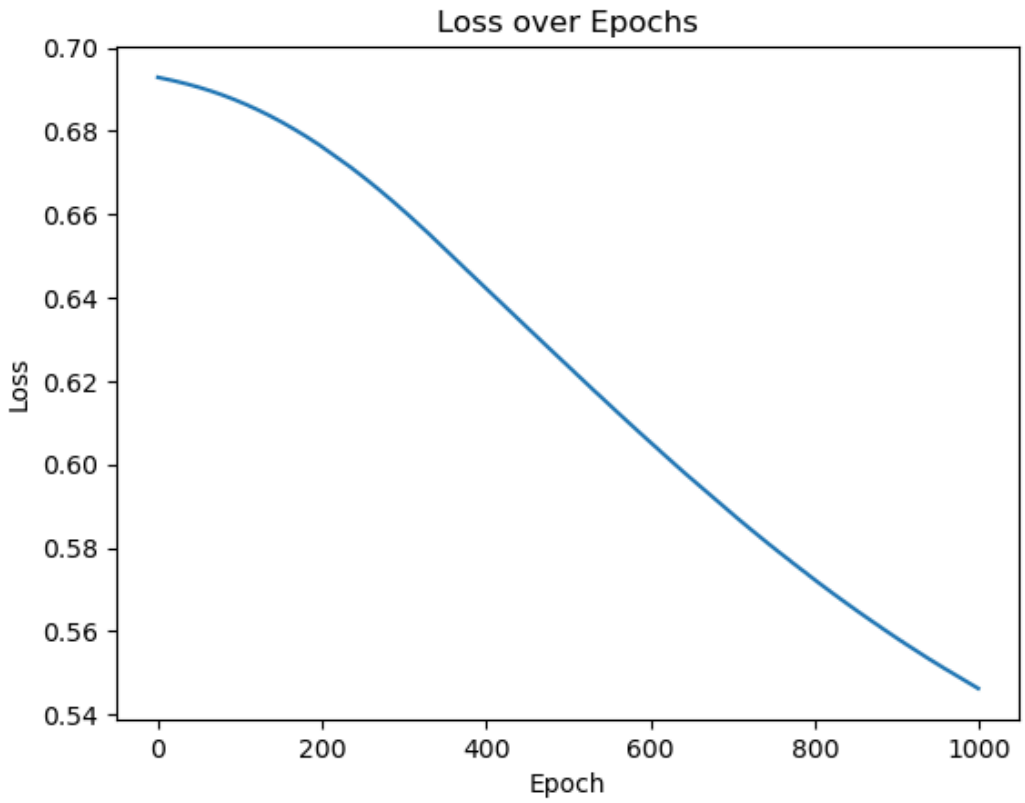
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

# 입력 데이터 생성
data = np.array([[0,0], [0,1], [1,0], [1,1]])
label = np.array([[0], [1], [1], [0]])

# 은닉층이 1개인 모델 생성
model = tf.keras.Sequential([
   tf.keras.layers.Dense(2, activation='relu', input_shape = (2,)),
   tf.keras.layers.Dense(1, activation='sigmoid')
])

# 모델 컴파일
model.compile(optimizer='adam',
            loss = 'binary_crossentropy',
            metrics=['accuracy'])

history = model.fit(data, label, epochs = 1000)

# 정확도 계산
print("Accuracy: ", model.evaluate(data, label)[1])

# 결정 경계를 위한 meshgrid
xx, yy = np.meshgrid(np.linspace(-0.1, 1.1, 300),
                     np.linspace(-0.1, 1.1, 300))
grid = np.c_[xx.ravel(), yy.ravel()]

# 각 데이터별 확률 출력
Z = model.predict(grid)
```

```
for i in range(4):
    print(f"{data[i]} → {grid[i][0]:.4f}")
```

위 OR 게이트 코드에서 입력 데이터만 바꿔 실행했다.

시각화





분리가 되지 않는 이유

XOR은 선형으로 분리가 불가능한 문제이기 때문에 하나의 퍼셉트론을 통해서는 분리할 수 없다.

하나의 perceptron은 $w1 * x1 + w2 * x2 + b = 0$ 인 선형 결정 경계를 학습하기 때문에 은닉층에 매우 많은 뉴런을 사용해 비선형성을 더해주고 활성화함수를 비선형 함수로 사용할 경우 이론상 가능하지만 적은 뉴런 개수로는 불가능하다.

XOR 게이트 학습 및 시각화 (DNN)

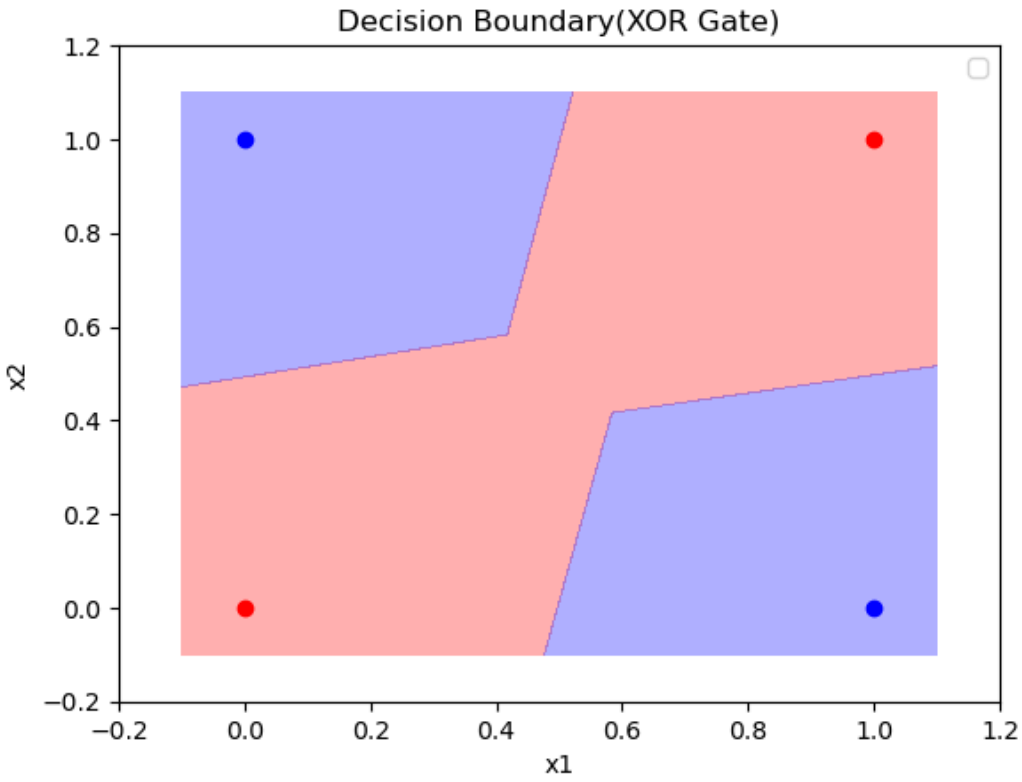
코드

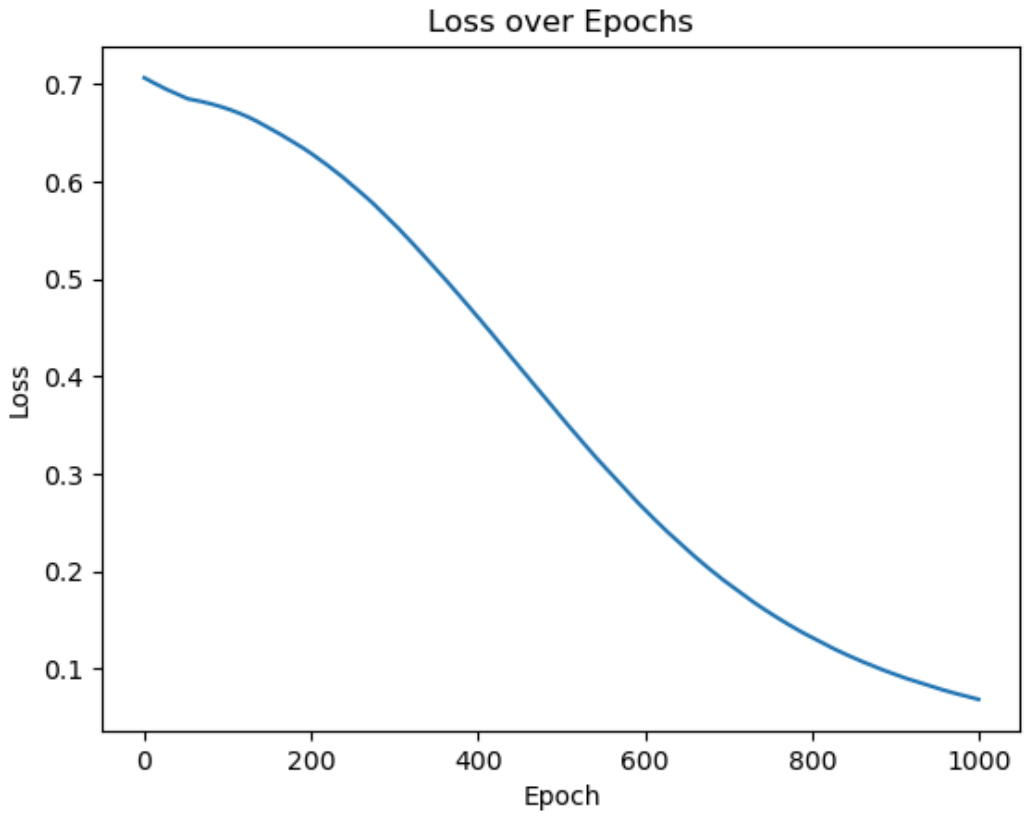
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

# 입력 데이터 생성
data = np.array([[0,0], [0,1], [1,0], [1,1]])
```

```
label = np.array([[0], [1], [1], [0]])

# 은닉층이 2개인 모델 생성
model = tf.keras.Sequential([
    tf.keras.layers.Dense(8, activation='relu', input_shape = (2,)),
    tf.keras.layers.Dense(4, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# 모델 컴파일
model.compile(optimizer='adam',
              loss = 'binary_crossentropy',
              metrics=['accuracy'])

history = model.fit(data, label, epochs = 1000)

# 정확도 계산
print("Accuracy: ", model.evaluate(data, label)[1])

# 결정 경계를 위한 meshgrid
xx, yy = np.meshgrid(np.linspace(-0.1, 1.1, 300),
                     np.linspace(-0.1, 1.1, 300))
grid = np.c_[xx.ravel(), yy.ravel()]

# 각 데이터별 확률 출력
Z = model.predict(grid)
for i in range(4):
    print(f"{data[i]} → {grid[i][0]:.4f}")
```

xor 게이트 코드와 같지만 출력이 각각 8과 4인 은닉층 2개를 추가했다.

시각화


Decision Boundary(XOR Gate)
x1
x2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Loss over Epochs
Loss
Epoch
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
200
400
600
800
1000

분리가 되는 이유

DNN은 ANN과 다르게 여러개의 은닉층과 퍼셉트론으로 비선형성을 계속 더해가며 표현 가능하기 때문에 XOR과 같이 선형적으로 분리되지 않는 문제도 표현할 수 있다.