```python
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
```

```python
# 1. 데이터 생성 (두 개의 클래스, 선형 분리 가능)
torch.manual_seed(42)
num_samples = 100
x0 = torch.randn(num_samples, 2) + torch.tensor([2.0, 2.0])   # 클래스 0
x1 = torch.randn(num_samples, 2) + torch.tensor([-2.0, -2.0]) # 클래스 1

X = torch.cat([x0, x1], dim=0)
y = torch.cat([torch.zeros(num_samples), torch.ones(num_samples)]).unsqueeze(1)  # (200,1)
```

```python
# 2. 모델 정의 (Logistic Regression)
class LogisticRegressionModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = nn.Linear(2, 1)  # 입력: 2차원, 출력: 1차원 (확률)

    def forward(self, x):
        return torch.sigmoid(self.linear(x))

model = LogisticRegressionModel()
```

```python
# 3. 손실 함수 & 옵티마이저
criterion = nn.BCELoss()  # Binary Cross Entropy Loss
optimizer = optim.SGD(model.parameters(), lr=0.1)
```

```python
# 4. 학습 루프
epochs = 100
for epoch in range(epochs):
    # Forward
    y_pred = model(X)
    loss = criterion(y_pred, y)

    # Backward
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch+1) % 10 == 0:
        print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")
```

```
Epoch 10/100, Loss: 0.2062
Epoch 20/100, Loss: 0.1167
Epoch 30/100, Loss: 0.0849
Epoch 40/100, Loss: 0.0682
Epoch 50/100, Loss: 0.0579
Epoch 60/100, Loss: 0.0508
Epoch 70/100, Loss: 0.0456
Epoch 80/100, Loss: 0.0415
Epoch 90/100, Loss: 0.0383
Epoch 100/100, Loss: 0.0357
```

```python
# 5. 결정 경계 시각화
def plot_decision_boundary(X, y, model):
    x_min, x_max = X[:,0].min()-1, X[:,0].max()+1
    y_min, y_max = X[:,1].min()-1, X[:,1].max()+1
    xx, yy = torch.meshgrid(torch.linspace(x_min, x_max, 100),
                            torch.linspace(y_min, y_max, 100))
    grid = torch.cat([xx.reshape(-1,1), yy.reshape(-1,1)], dim=1)
    with torch.no_grad():
        probs = model(grid).reshape(xx.shape)
    plt.contourf(xx, yy, probs, levels=[0,0.5,1], alpha=0.3, colors=['blue','red'])
    plt.scatter(X[:,0], X[:,1], c=y.squeeze(), cmap="bwr", edgecolors='k')
    plt.show()
```

```python
plot_decision_boundary(X, y, model)
```